Departamento de Teoría de la Señal y Comunicación

TRABAJO FIN DE GRADO

Detección de la saliencia visual dinámica: hacia el modelado de la atención

Autora: Laura López Aragoneses

Tutora: Carmen Peláez Moreno

Leganés, junio de 2016

Título: "Detección de la saliencia visual dinámica: hacia el modelado de la atención"
Autora: Laura López Aragoneses
Directora: Carmen Peláez Moreno

EL TRIBUNAL

Presidente: Pablo Acedo Gallardo

Vocal: Pablo Serrano Yañez-Mingot

Secretario: Matilde Pilar Sánchez Fernández

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 5 de julio de 2016 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

En primer lugar, me gustaría darle las gracias a mi, Carmen Peláez Moreno, que tanto me ha ayudado durante estos meses.
Quiero darle las gracias a mis padres por hacer todo lo posible para que haya llegado hasta aquí y a mis hermanos que han guiado mis pasos.
A mis amigas, Andrea y Jennifer, que me han ayudado durante todo este tiempo.
A mis compañeros de universidad, en especial a Javier, que ha sido mi compañero de laboratorio y de risas durante todos estos años.
Y finalmente, gracias a Jesús Manuel, por apoyarme siempre y darme ánimos cuando creía que nada iba a salir bien.

Índice general

Índice de figuras

Índice de tablas

Chapter 1

Introduction and objectives

1.1 Introduction

The evaluation of the features of an image analysing the eye movements is important for developing computer vision applications, as well as the understanding of how biological systems explore the environment. For this reason, there are numerous models that seek to predict where the human visual attention will be focused when watching an image, i.e. its visual saliency.

In this project, a dynamic saliency model has been modified towards the attention modeling. This has been done by adding a face detector and a center-bias to an existing saliency model.

We will start explaining some saliency and face detection algorithms, delving into the ones that have been used in order to carry out our objective. Then, the description of the developed model is given, together with the decisions that have been taken. Finally, we included some experiments and its results to evaluate the performance of the implemented system.

A budget showing the costs of improving the algorithm is given in chapter 6.

1.2 Objectives

Nowadays, most saliency models are focused on the extraction of bottom-up information, like color, contrast or motion, to predict the eye fixations of an observer. Nevertheless, human visual attention focuses on high-level features of the image, which provide relevant information in order to understand the scene –top-down attention–.

A small amount of visual saliency algorithms include this top-down attention, so our purpose is to improve the performance of an existing bottom-up saliency model, by adding some high-level features: the tendency of looking to the center of the screen and the fact that we pay more attention to faces. These two top-down cues were included, although the proposed problem was to implement only the existence of faces.

In order to build this hybrid model, we are going to start from a dynamic bottom-up saliency algorithm, developed by Carlos Ruiz in his Final Year Project[46] (in Spanish, *Trabajo Fin de Grado*, TFG) –explained in section 3.1–. So, the main objective is to improve its performance. The obtained results and the comparison between both models can be seen in section 4.4.

On the other hand, enlarging the eye tracking database was proposed as future work in his TFG, so other objective is to accomplish this task. Although a large database formed by 60 videos with eye fixations data was found, only 8 of them have been used due to the time restrictions, as it is detailed in section 4.1.

1.3 Stages of development and material used

To achieve the objectives, some steps have been followed. They are briefly explained below, together with the elements used in each one of them.

1. **Initial approach:** the purpose of this first step was to understand the task to be tackled and delve into it. In addition, the first decisions on the algorithms that have been used were taken.

2. **Algorithm development:** this phase can be divided into three steps.

    a. Dynamic saliency software adjustment: the saliency software[46] that has been improved, reads the eye fixations data from a *.xml* file. However, the eye tracking database[1] that have been used was built in Matlab, so the software had to be adapted.

    b. Incorporation of top-down cues: two important characteristics of human visual attention have been included in the saliency algorithm, as it was mentioned above: the tendency of looking to the center of the screen and the fact that we pay more attention to faces. Both have

[1] Database 1, *Dynamic natural scenes*: http://antoinecoutrot.magix.net/public/databases.html

been developed in Matlab, the first by introducing a bias proportional to a Gaussian distribution, situated in the center of the images (section 3.2.2) and the second using the Viola-Jones face detection algorithm, implemented in Matlab's *Computer Vision System Toolbox*[2] (section 3.2.1).

c. Neural network training: firstly, the software to accomplish this task was chosen. In this case, the Matlab's *Neural Network Toolbox*[3] has been used (section 3.2.3). Then, various training sets have been built as is explained in 4.2, in order to find the best way to train the network.

3. **Algorithm evaluation:** to evaluate the modified saliency model, two videos have been tested with different training sets. The results can be seen in section 4.4.

4. **Report writing:** this last step has been carried out by looking up several research papers, listed in the references section. Microsoft Word has been used as an editing tool.

The sequence of tasks and the time spent in its execution is detailed in section 6.1, where a Gantt chart is also included.

1.4 Socio-economic environment

The prediction of eye movements has a considerable number of applications, resulting in a great socio-economic impact. For this reason, the aim of this project is to improve the results obtained by Carlos Ruiz, adding top-down visual features to his saliency model. Some of these applications are going to be mentioned below, although there are many more.

This system can be used in marketing in two ways. Firstly, it is useful to analyse in which parts of a television commercial, are the ones where viewers will focus their attention, to allow a better design. Secondly, a dynamic saliency algorithm can help improve a store's products layout system. To do this, the possible path followed by the customer is recorded in first person, in order to know the highly visible areas. Then, the placement of items can be corrected according to those results.

The saliency model also has medical purposes. For example, if it worked in real time, it could be used to detect tumors by making an ultrasound scan. Recently, automatic tumor detection in ultrasound images has been studied by Shao et al.[51], so it would be possible to extend this to ultrasound videos.

Some other applications that have been previously studied are mentioned by Borji and Itti[5], so they can be enhanced by this new dynamic saliency algorithm: video

[2] *Computer Vision System Toolbox*: http://es.mathworks.com/products/computer-vision/
[3] *Neural Network Toolbox*: http://es.mathworks.com/products/neural-network/

compression (Ourhani et al., 2003; Itti, 2004; Guo y Zhang, 2010), video summarization (Marat et al., 2007; Ma et al., 2005), dynamic lighting (Seif El Nasr, 2009), video shot detection (Boccignone et al., 2005) and many more.

All of these applications represent a great change in everyday life, as they can be applied to social networks, television, medicine, etc.

Finally, it is important to mention that a budget showing the costs of improving the algorithm developed by Carlos Ruiz has been included in 6.2.

1.5 Regulatory framework

Two main issues are discussed in this project: visual saliency models and face detection, but neither has a regulatory framework yet. Nevertheless, both of them are used in numerous applications, so some of these will be subject to regulation, for example facial recognition.

Facial recognition is one of the most important fields in which face detection is applied, as a first step to locate the face that has to be identified. For example, social networks have introduced this technology in photo tagging. It is also used in authentication and security, even in public places. But one question may come to mind: what about our privacy? This is a common question when talking about facial recognition that does not have a clear answer. In some countries, like Spain[4], this issue is included in the data protection law.

In the future, it is unlikely that neither face detection nor saliency models have a general regulatory framework, due to the difficulties to cover all of their applications. Some of these will be regulated separately, instead.

[4] Reports: 0392/2011 and 0328/2012

Capítulo 2

Estado del arte

Evaluar las propiedades de una imagen analizando dónde se sitúa la mirada de una persona, es importante tanto para el desarrollo de aplicaciones de visión artificial, como para entender cómo los sistemas biológicos exploran el entorno. Hay una gran cantidad de modelos que persiguen predecir a qué puntos de una imagen se dirigirá el ojo humano es decir, su saliencia visual[28].

El campo de la predicción de movimientos oculares, que comenzó con el influyente modelo de Itti et al.[22], cuenta ahora con numerosos modelos predictivos de los cuales solo unos pocos han incorporado el tratamiento de la atención *top-down*[28]. De esta manera, existen modelos de saliencia visual *bottom-up*, atención *top-down* y una mezcla de ambos. Los primeros están basados en características visuales de la imagen y son los más abundantes, mientras que los segundos están determinados por fenómenos cognitivos que permiten la identificación de objetos y escenas[5].

La atención *bottom-up* es rápida e involuntaria y se ve atraída por regiones notablemente diferenciadas del entorno que las rodean. En contrapartida, la atención *top-down* es lenta y voluntaria y varía según el próposito del individuo que visualiza la escena. Por esta razón, es difícil llevar a cabo un modelo generalizado, de manera que tres cuestiones han sido principalmente exploradas: cómo decidimos dónde mirar, cómo afecta el contexto de la escena a las fijaciones y dónde miramos exactamente[5].

A pesar de que muchos modelos están clasificados dentro de la categoría *bottom-up*, la mayoría de los movimientos oculares están impulsados por características de alto nivel[18]. Además, el progreso de los algoritmos de saliencia visual es medido mediante la comparación de la información capturada por cada uno, suponiendo que el mejor modelo

será el que obtenga una mayor cantidad de información relevante para el comportamiento humano[28].

Por estos motivos se ha decidido modificar el algoritmo de detección espacio-temporal propuesto por Carlos Ruiz[46], introduciendo dos aspectos de la atención *top-down*: la tendencia de los observadores a mirar al centro de las imágenes, explicada con más detalle en la sección 3.2.2, y la tendencia a fijar la atención en los rostros, que se conseguirá mediante un detector facial.

En este capítulo se van a exponer, en primer lugar, las diferentes aproximaciones para llevar a cabo la detección de saliencia visual, tanto estática como dinámica, haciendo hincapié en los modelos empleados en el sistema de saliencia espacio-temporal que se va a modificar. En segundo lugar, se explicarán algunos de los numerosos algoritmos existentes para la detección facial, prestando especial atención al método empleado, el de Viola y Jones[55]. Se diferenciarán aquellos basados en plantillas rígidas y dinámicas de los desarrollados mediante redes neuronales. Finalmente se mencionarán brevemente otros modelos.

2.1 Saliencia visual

En la actualidad existen tanto modelos de saliencia estática o espacial, como modelos de saliencia dínamica o espacio-temporal. Los primeros son más numerosos y están basados en la extracción de las regiones de interés de una imagen fija, dando lugar al mapa de saliencia estática. Los segundos también tienen en cuenta el movimiento, por lo que combinan el mapa de saliencia estática con el de saliencia dinámica, para dar lugar a un único mapa de saliencia.

Durante esta sección se darán unas breves pinceladas sobre diferentes algoritmos de ambos tipos de saliencia, ya que esta clasificación fue expuesta con mayor detalle en el TFG de Carlos Ruiz[46].

2.1.1 Saliencia estática

Desde que Koch y Ullman[27] definieron en 1985 el concepto de mapa de saliencia, se han sucedido numerosos algoritmos de saliencia estática orientados tanto a la saliencia *bottom-up*, como a la atención *top-down*. En este apartado se van a enumerar algunos de ellos, profundizando en el de Itti et al. [22], en el que está basado el modelo de saliencia empleado en este trabajo.

El algoritmo de saliencia estática más famoso es el definido por Itti et al.[22] y está basado en la saliencia *bottom-up*. A continuación se va a explicar su funcionamiento, diferenciando cuatro pasos: extracción de características, cálculo de los mapas de características, extracción del mapa de saliencia y detección de la región saliente.

En este modelo, la imagen de entrada se escala en nueve tamaños diferentes, con factor de reducción desde 1 hasta 8[3], generando una pirámide gaussiana. Posteriormente

se extrae información de cada nivel sobre la variación de intensidad, de color y de orientación, y se calculan las diferencias *center-surround*. De esta manera, se crea un mapa de características para cada escalón de la pirámide.

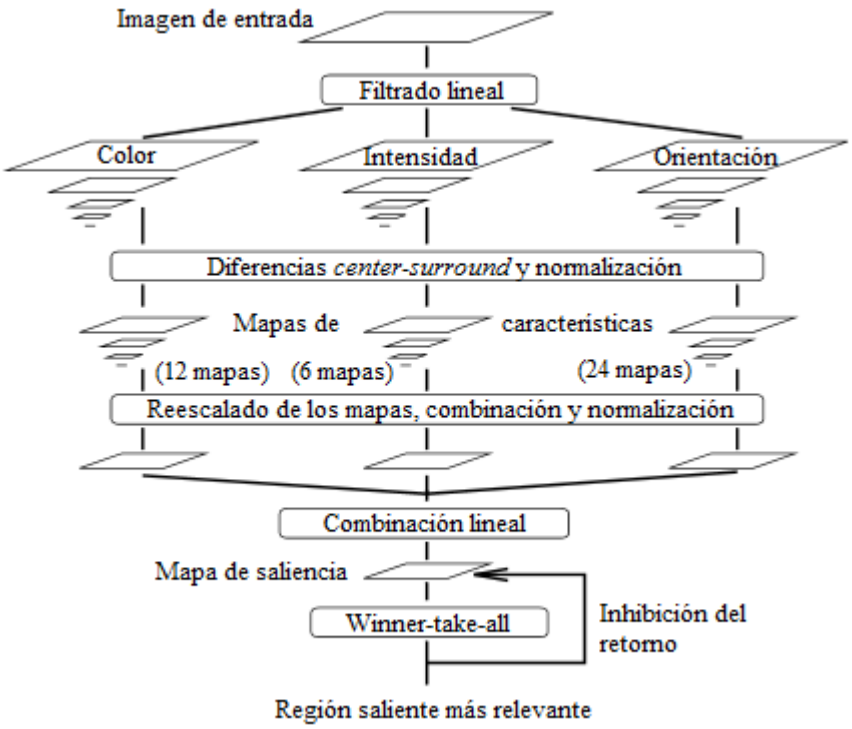


Figura 1. *Arquitectura del modelo de saliencia de Itti et al.*[22].

Una vez hecho esto, se normalizan los mapas de modo que sobresalgan los que tengan altos valores máximos y se de menos importancia a los que presentan pocas variaciones. A continuación, se reescalan todos los mapas al nivel inferior de la píramide y se suman, formando un solo mapa por cada característica (intensidad, color y orientación).

Finalmente, los mapas se combinan linealmente con el mismo peso para dar lugar a un único mapa de saliencia. Para detectar la zona del mapa donde el estímulo de saliencia es más relevante, se emplea un modelo de red neuronal basado en el principio *winner-take-all*.

Por otro lado, Liu et al.[35] propusieron una mejora de este modelo, basado en una única característica: el contraste de la imagen. También introdujeron la segmentación de la imagen, con el fin de determinar la saliencia de una región como la media de la saliencia de los píxeles pertenecientes a ésta. El algoritmo de saliencia dinámica empleado en este trabajo, también hace uso de la segmentación de la imagen, como se verá más detalladamente en la sección 3.1.3.

Achanta et al.[1] llevaron a cabo un modelo de saliencia estática basado en la diferencia de gaussianas (DoG), que extrae dos características de las imágenes, el color y la luminancia. Su objetivo era mejorar los resultados de otros algoritmos en tres aspectos: resolución de los mapas de saliencia, definición uniforme de las regiones salientes con bordes bien definidos y coste computacional.

Existen otros modelos de saliencia estática que incluyen la atención *top-down*, como AIM (de su nombre en inglés, *Attention based on Information Maximization*) desarrollado por Bruce y Tsotsos[7], DVA (de su nombre en inglés, *Dynamic Visual Attention*) definido por Hou y Zhang[20] y SUN (*Saliency Using Natural statistics*), un algoritmo de Zhang et al.[63].

2.1.2 Saliencia dinámica

El sistema visual humano descarta parte de la información recibida cuando es estimulado, para atender a los objetos de interés. Cuando ese estímulo es un vídeo y no una imagen estática, estimar la saliencia se convierte en un problema más complicado. Esto se debe a que la atención puede variar a medida que el vídeo avanza, y lo que podía ser saliente en un principio, se vuelve insignificante para el observador una vez que ya ha sido explorado.

Mientras que la saliencia estática ha sido estudiada en profundidad, pocas investigaciones han indagado en la saliencia dinámica de secuencias de vídeo, debido a su complejidad. La aparición de bases de datos de vídeos con seguimiento ocular durante los últimos tiempos, ha facilitado este trabajo.

El cálculo del flujo óptico, es el método más empleado por los algoritmos de saliencia dinámica existentes, que extrae el movimiento entre cada par de *frames* consecutivos para dar lugar al mapa de saliencia temporal. La primera definición del flujo óptico fue realizada por Horn y Schunck en 1981[19], la cual fue mejorada por Black y Anandan[4].

El modelo de saliencia espacio-temporal elegido[46], explicado con detalle en la sección 3.1, hace uso del modelo de Liu[34] para el cálculo del flujo óptico. Éste está basado en los trabajos de Brox et al.[6] y Bruhn et al.[8], aunque tiene una gran diferencia, y es que Liu emplea el método del gradiente conjugado para resolver sistemas lineales, de manera que la programación del algoritmo es más sencilla.

En 2016 un nuevo modelo espacio-temporal de análisis de atención, fue propuesto por Zhong et al.[64] denominado STAM, en el que se distinguen tres partes: detección de saliencia espacial, temporal y una fusión de ambos mapas de saliencia. En la parte de extracción de características perteneciente a la saliencia espacial, se obtienen la intensidad, el color, la orientación y el contraste. También se introduce una nueva técnica (en ingles, V*isual Orientation Inhomogeneous Saliency*, VOIS) que define un mapa de orientación de características diferente al original, que incluye orientaciones cardinales. A la hora de calcular el mapa de saliencia temporal, se emplea un modelo de flujo óptico que estima el movimiento teniendo en cuenta su consistencia en secuencias de vídeo (en inglés, *Dynamic Consistent Optical Flow, DCOF*). Para mezclar ambos mapas de saliencia, se emplea el método *skew-max*.

2.2 Detección facial

En la década de los setenta surgieron los primeros sistemas de detección facial, pero fue en los noventa cuando se produjeron aportaciones significativas en este campo, gracias al desarrollo tecnológico. Sin embargo, detectar una cara correctamente no es tarea fácil, ya que hay que tener en cuenta numerosos factores como las condiciones de iluminación, la posición de la cara o las expresiones faciales[59].

Para introducir la existencia de caras en un fotograma como característica de alto nivel, como se justifica en la introducción de este capítulo, se ha hecho uso de un detector facial cuya implementación se detalla en la sección 3.2.1. En este apartado, se va a realizar una clasificación de diferentes modelos en tres grupos: basados en plantillas rígidas, deformables y los que hacen uso de redes neuronales. Finalmente se van a enumerar algunos otros algoritmos de detección facial.

2.2.1 Algortimos de detección facial basados en plantillas rígidas

Estos algoritmos están basados en el aprendizaje de plantillas rígidas empleando cascadas de clasificadores. A éstos se les aplica una técnica de *boosting*, que consiste en la construcción de clasificadores complejos a partir de una serie de clasificadores básicos[36]. Dos componentes clave de estos algoritmos, son la extracción de características y la elección del algoritmo de aprendizaje. En este apartado se van a exponer varios algoritmos de detección facial, haciendo hincapié en el que se ha empleado, el desarrollado por Viola y Jones.

Sakai et al.[47] llevaron a cabo un modelo de detección facial frontal, en el que emplearon múltiples plantillas para detectar las diferentes partes de un rostro, como son los ojos o la boca, y así modelarlo. Este algoritmo se divide en dos fases: una para encontrar las regiones de interés de la imagen y otra para determinar la existencia de una cara en esas regiones.

Craw et al.[13] sin embargo, propusieron un método en el que primero se buscan los bordes de la imagen con un filtro de Sobel, para después compararlos con una plantilla que posee la forma de una cara. A continuación, se diferencian el resto de rasgos faciales haciendo uso de otras plantillas. Un sistema similar a éste es el de Govindaraju et al.[16], que también obtiene los bordes de la imagen, pero esta vez mediante el operador de Marr-Hildreth, para después aplicar un filtro que elimina las formas que no pertenecen a un rostro.

Otros algoritmos de este tipo son los de Tsukamoto et al.[53], en el que la imagen es dividida en bloques, de los que se extrae características para comprobar si existe una cara, y Samal et al.[48], que publicaron un modelo basado en plantillas de la silueta de un rostro, obtenidas mediante PCA.

2.2.1.1 Algortimo de Viola y Jones

Este algoritmo supone el primer sistema de detección facial en tiempo real de la historia[57]. Está formado por tres elementos: el cálculo rápido de características mediante una representación de la imagen denominada *imagen integral* (en inglés, *integral image*), la clasificación de éstas con el algoritmo de aprendizaje AdaBoost y un método que combina clasificadores en cascada, encargado de descartar regiones de la imagen no prometedoras para la detección facial, aumentando considerablemente la velocidad del algoritmo.

De acuerdo al artículo presentado en el año 2004 por Viola y Jones[55], su algoritmo basado en la detección frontal de caras, tiene unas prestaciones similares a los mejores resultados publicados hasta la fecha de su creación (Sung y Poggio, 1998[52]; Rowley et al., 1998[45]; Osuna et al., 1997[38]; Schneiderman y Kanade, 2000[49]; Roth et al., 2000[44]), pero introduce una notable mejora: su velocidad es superior a la de cualquier otro sistema.

Extracción de características

Los autores de este algoritmo se vieron motivados por el trabajo de Papageorgiou et al.[39], que emplearon un conjunto de funciones base de tipo Haar para obtener diferentes características. En concreto, Viola y Jones utilizaron tres tipos: con dos, tres y cuatro rectángulos (véase Figura 2).

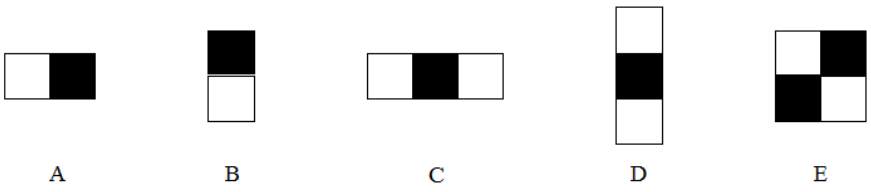


Figura 2. *Diferentes tipos de plantillas[23]. (A) y (B) muestran las plantillas de dos rectángulos, (C) y (D) las de tres y (E) la de cuatro.*

Para obtener el valor de la característica, se sitúa una función base sobre una región de la imagen, y se sustrae la suma de todos los píxeles pertenecientes a los rectángulos blancos, a la suma de los que se encuentran bajo rectángulos negros.

En su artículo[55], Viola y Jones afirman que un detector con una base de resolución de 24x24 píxeles, puede obtener más de 160.000 características diferentes, por lo que éstas deben ser calculadas rápidamente. Para este fin introdujeron una nueva manera de representar una imagen, denominada imagen integral, que permite una rápida extracción de estas características.

1	1	1
1	1	1
1	1	1

Imagen de entrada

1	2	3
2	4	6
3	6	9

Imagen integral

Figura 3. *Cálculo de la imagen integral[23].*

En la imagen integral, cada píxel es la suma de todos los píxeles que se encuentran encima y a la izquierda, como se demuestra en la Figura 3. Esto permite sumar los píxeles en el interior de cualquier rectángulo a partir de cuatro valores, que coinciden con los píxeles de la imagen integral correspondientes a los cuatro vértices del rectángulo (véase Figura 4).

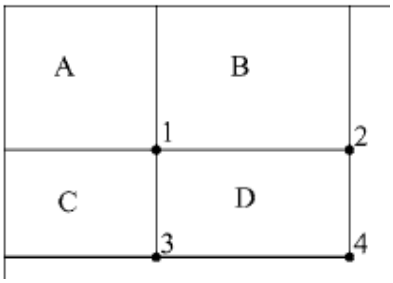


Figura 4. *La suma de los píxeles en el interior del rectángulo D puede calcularse a través de los cuatro puntos numerados[55]. Se puede ver que el valor del píxel situado en la posición 1, es la suma de los píxeles en el rectángulo A, el valor en la posición 2 es A + B, en la posición 3 es A + C y en 4 es A + B + C + D. Por tanto, la suma de los píxeles situados en D se calcula como 4 + 1 – (2 + 3).*

Algoritmo de aprendizaje AdaBoost

Como se mencionó anteriormente, hay más de 160.000 características asociadas a cada región de una imagen, por lo que procesar todo este conjunto supondría un coste computacional muy elevado. Por esta razón, Viola y Jones introdujeron en su sistema una variante de AdaBoost (Freund y Schapire, 1995[49]), con el fin de seleccionar las características más importantes.

AdaBoost es una técnica *boosting* de aprendizaje que combina una serie de clasificadores débiles, a los que se asignan diferentes pesos. A las muestras correctamente clasificadas se les da menor importancia que a las que han sido clasificadas erróneamente, de manera que los próximos clasificadores débiles se centren en estas últimas.

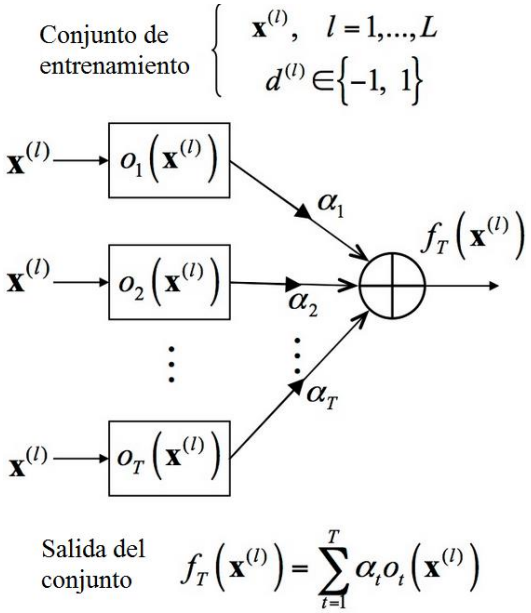


Figura 5. *Esquema del algoritmo AdaBoost[54].*

Clasificadores en cascada

Esta parte es la encargada de reducir notablemente el tiempo de procesamiento, además de aumentar las prestaciones del detector. Para ello, se emplean clasificadores simples que descartan la mayor parte de regiones de la imagen, antes de que lleguen a los más complejos y se produzcan falsos positivos.

Esta cascada se construye mediante el entrenamiento de clasificadores con AdaBoost, y está compuesta por clasificadores duros de dos características (véase Figura 2 (A) y (B)). Así, se reduce fácilmente el número de subventanas que requieren un procesado posterior. En la Figura 6 se puede ver este proceso, en el que un resultado positivo activa el siguiente clasificador, mientras que uno negativo hace que se descarte la región evaluada.

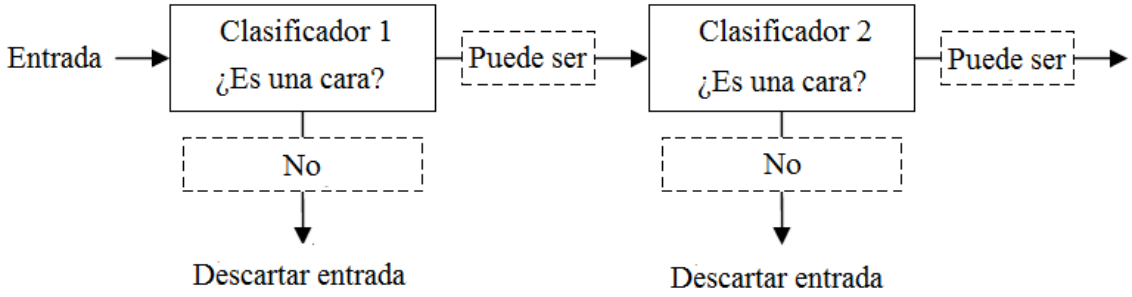


Figura 6. *Clasificador en cascada*[23].

Como consecuencia, la cantidad de falsos negativos que se espera tener al final de la cascada es baja, por lo que este algoritmo de detección facial proporciona altas prestaciones en un tiempo reducido.

2.2.2 Algoritmos de detección facial basados en plantillas deformables

En este apartado se va a exponer otro tipo de detección facial basado en plantillas, pero esta vez deformables, es decir que cambian sus características en función de las imágenes procesadas, como explican Yang et al.[59] en su artículo.

Los modelos de partes deformables (en inglés, *Deformable Parts Models*, DPMs) son generalmente utilizados en la detección de objetos. Los primeros en hacer uso de este modelo para la detección facial fueron Yuille et al.[44], que plantearon un nuevo método en el que las características faciales se detectan a través de plantillas deformables, explicado a continuación.

En este sistema, las plantillas cambian su tamaño así como muchos otros parámetros, para adaptarse a los datos de entrada, por lo que el algoritmo debe funcionar a pesar de variaciones de iluminación, posición de la cara o cambios de *zoom*. Por otro lado, se define una función de energía que es mínima donde mejor se ajusta a la imagen y se actualizan los valores de los parámetros de las plantillas. Esto hace que tanto la posición como las características de éstas cambien, de modo que los parámetros iniciales, determinados mediante preprocesado, son muy diferentes de los finales.

Otros métodos que emplean plantillas deformables, son los diseñados por Kwon[29] y Lam et al.[30], que hacen uso de *snakes* para localizar el contorno de la cara. Éstas evolucionan mediante la minimización de una función de energía, igual que en el caso expuesto anteriormente. A continuación, se aplican plantillas deformables para la búsqueda de rasgos faciales.

Lanitis et al.[31] describieron un método de representación facial que aporta información de la forma y la intensidad. El modelo de plantillas deformables que emplean se denomina *Point Distribution Model* (PDM) y se encarga de detectar automáticamente las características faciales, así como de clasificar los rostros en función de su forma. Otros algoritmos similares son los de Kirby et al[26]. y Cootes et al.[11].

2.2.3 Modelos de detección facial con redes neuronales

Las redes neuronales han sido comúnmente empleadas para realizar detectores faciales, como los propuestos en un inicio por Rowley et al.[45] y Roth et al.[44]. Más tarde, aparecieron otros detectores basados en diferentes tipos de redes neuronales.

En 2001 Féraund et al.[14], desarrollaron un detector facial que determina si una región de la imagen de 15x20 píxeles, contiene una cara o no, para lo que se emplean cuatro elementos. El primero es un filtro de movimiento, en el caso de análisis de secuencias de vídeo; a continuación se aplica un filtro del color de la piel, en el caso de imágenes a color; un perceptrón multicapa y un sistema basado en la combinación de un modelo de red neuronal llamado *Constrained Generative Model* (CGM), empleado para la detección de caras vistas desde diferentes ángulos.

En el año 2007, Osadchy et al.[37] diseñaron un método para detectar caras y estimar su posición en tiempo real, haciendo uso de una red neuronal convolucional, como se puede observar en la Figura 7, cuya entrada es una imagen de 32x32 píxeles en escala de grises.

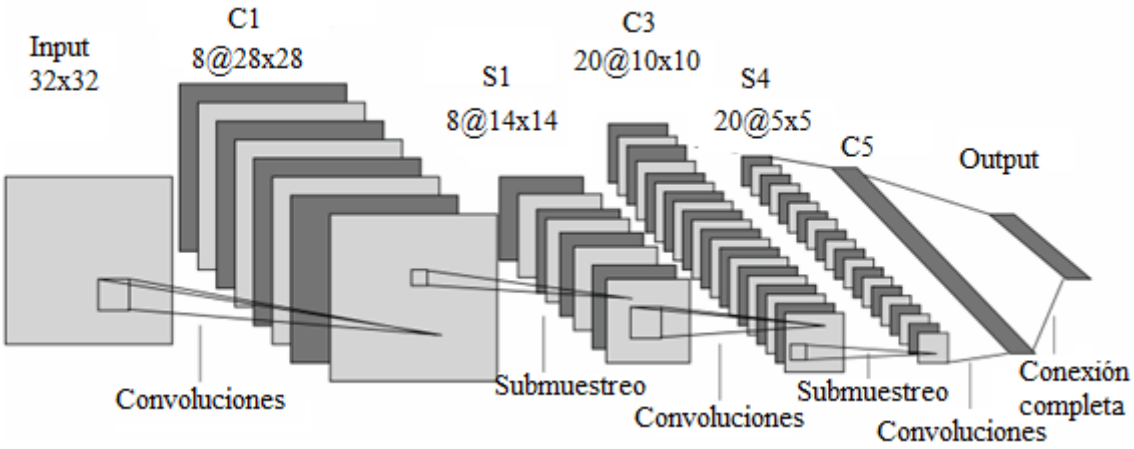


Figura 7. *Arquitectura de la red neuronal convolucional empleada por Osadchy et al[37].*

Chen et al.[9] llevaron a cabo en 2009 un detector facial con dos partes: un detector inicial poco preciso y otro con mayor precisión. Este último está formado por una red neuronal muy sencilla, cuyas entradas son características extraídas a través de

transformaciones de Haar. Esta red tiene 457 parámetros, un número muy pequeño si se compara con la cantidad de parámetros empleados por Osadchy et al. (63.493).

Durante estos últimos años, se ha comenzado a emplear redes neuronales profundas convolucionales (en inglés, *Deep Convolutional Neural Networks*, DCNNs) para la detección de objetos[15]. En 2014, Zhang et al.[62] desarrollaron un detector facial capaz de detectar caras en cualquier posición, entrenando la DCNN para que realizara tres tareas: detección del rostro, posición del mismo y localización de puntos de referencia.

2.2.4 Otros algoritmos de detección facial

Además de los vistos anteriormente, existen otros algoritmos que también proporcionan buenos resultados[61]. A continuación, se van a mencionar algunos de ellos.

El método *Antifaces* propuesto por Keren et al.[25] emplea una secuencia de detectores sencillos, en la que solamente las imágenes que superan un umbral pasan al siguiente detector. El rendimiento de este algoritmo es comparable con los basados en autocaras y en máquinas de soporte vectorial (en inglés, *Support Vector Machines*, SVMs), pero su velocidad es mayor.

Por otro lado, Liu[33] llevó a cabo un modelo de detección facial frontal denominado *Bayesian Discriminating Features* (BDF) que introduce tres factores novedosos: el análisis discriminativo de la imagen de entrada; el modelado estadístico de las imágenes que contienen caras y las que no, y un clasificador bayesiano para la detección de múltiples rostros en posición frontal.

Hay numerosos detectores faciales que emplean máquinas de soporte vectorial, como el presentado por Osuna et al.[38], debido a sus altas prestaciones en problemas de aprendizaje automático, pero la velocidad de detección con este método es baja. Así, diferentes soluciones han sido propuestas, como las de Romdhani et al.[43] y Rätsch et al.[42]. Además, las SVMs también han sido empleadas en la detección facial multivista por Li et al.[32] y Yan[58].

Capítulo 3

Descripción del modelo

El software de saliencia espacio-temporal desarrollado por Carlos Ruiz[46] consta de tres partes: el cálculo de la saliencia estática, el cálculo del flujo óptico y la segmentación de la imagen. De este modo, extrae tres descriptores de todos los píxeles de cada fotograma: saliencia estática, velocidad y aceleración.

Éste va a ser modificado añadiendo dos descriptores más, como ya se dijo anteriormente, con el fin de modelar de alguna manera la atención visual humana: uno indicando en qué píxeles de la imagen existe una cara y otro que resalta la importancia del centro de la imagen.

Después, se entrena una red neuronal del tipo perceptrón multicapa (en inglés, *MultiLayer Perceptron*, MLP), que se encargará de clasificar los píxeles de todos los *frames* de un nuevo vídeo, como salientes o no salientes. Para esto, la red neuronal necesita dos elementos: unas entradas que serán los descriptores extraídos, y un objetivo, o salidas, obtenido mediante un *eye-tracker*, que indican la saliencia de las regiones de la imagen.

Un *eye-tracker* es un equipo que extrae las coordenadas del fotograma a las que el sujeto del experimento ha mirado (fijaciones oculares). De esta manera, en el modelo de saliencia dinámica empleado, se construye una matriz binaria para cada *frame*, en la que los unos representan una región circular de radio variable, como se verá en el capítulo de experimentos, cuyo centro está situado en las coordenadas extraídas por el *eye-tracker*. Así, esta matriz será el objetivo de la red neuronal.

En este apartado se van a explicar las tres partes de las que consta el modelo de saliencia dinámica, además de las modificaciones realizadas para introducir los nuevos

descriptores a parte de los ya existentes. Ambos serán evaluados en el capítulo 4, donde se decidirá si son empleados o se descarta alguno de los ellos.

Por último, se va a exponer qué software se ha elegido para el entrenamiento de la red neuronal, mencionada anteriormente.

3.1 Modelo de saliencia dinámica

Aunque escasos, existen diferentes sistemas de detección de saliencia espacio-temporal, como los que se han mencionado en el apartado 2.1.2. En este caso, se ha empleado un algortimo desarrollado en Matlab para un Trabajo Fin de Grado anterior[46], debido a su disponibilidad. A continuación, se van a explicar las tres partes de las que está compuesto este modelo, justificando por qué su autor hizo uso de determinados elementos, siempre teniendo en cuenta que su objetivo era desarrollar un modelo en Matlab que superara a los mejores algoritmos de saliencia estática.

3.1.1 Cálculo de la saliencia espacial

Para llevar a cabo el cálculo de la saliencia estática se ha hecho uso de la *SaliencyToolbox* de Walther y Koch[56], que está basada en el modelo de Itti et al.[22] expuesto en la sección 2.1.1. Se barajó entre distintas opciones como se puede ver en la Tabla 1, pero finalmente fue elegida la *SaliencyToolbox* gracias a su implementación en Matlab, su sencilla interfaz gráfica y sus buenas prestaciones.

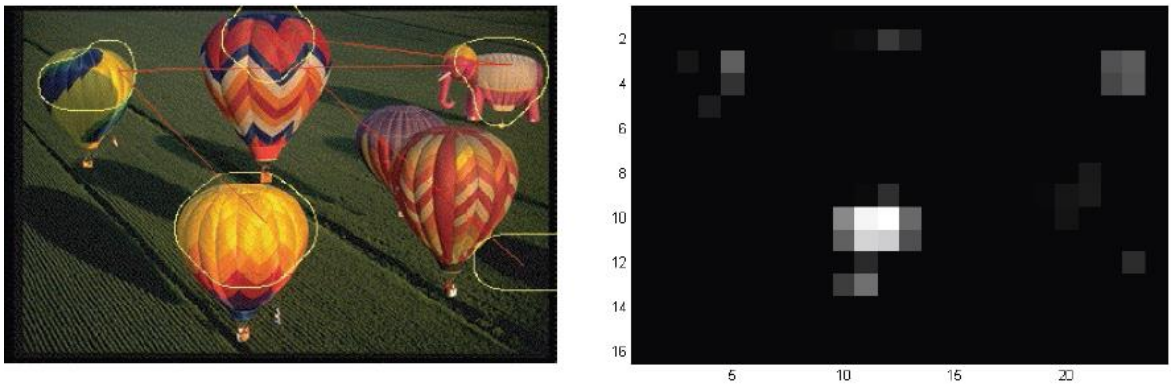
Autor(es)	Lenguaje
Cheng et al.[10]	C++
Perazzi et al.[40]	C++
Judd et al.[24]	Matlab
Hou y Zhang[21]	Matlab
Walther y Koch[56]	Matlab
Itti y Koch[22]	C++

Tabla 1. *Datos sobre varias implementaciones de modelos de saliencia estática.*

Esta *toolbox* es altamente configurable a través de su interfaz gráfica y consta de diferentes parámetros, de los cuales solamente fueron modificados los niveles de la pirámide gaussiana y el número de iteraciones de normalización de los mapas de características, con el fin de obtener unos mejores resultados.

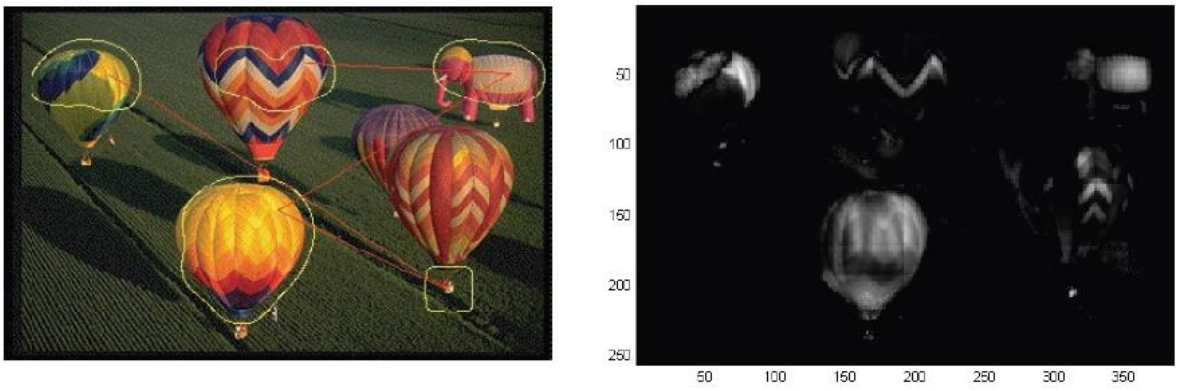
El valor del parámetro que indica a qué nivel de la pirámide se calcula el mapa final de saliencia, es 5 por defecto. Esto hace que el mapa de saliencia tenga unas dimensiones menores que la imagen original, por lo que fue modificado a 1 para que éste tuviera la misma resolución. En cuanto al número de iteraciones de normalización, cuanto mayor es este parámetro, menor detalle se obtiene en el mapa de saliencia, por lo que se cambió su valor por defecto (en este caso 3 iteraciones), a 1.

En las siguientes figuras se puede apreciar el mapa de saliencia obtenido con los parámetros por defecto y el resultante de establecer los valores anteriormente mencionados. De esta manera se observa que, mediante esta modificación, se ha cumplido el objetivo de mejorar los resultados en concordancia a las necesidades del problema.



A. 5 regiones más salientes **B.** Mapa de saliencia

Figura 8*. Resultados de la SaliencyToolbox con los parámetros por defecto (resolución del mapa de saliencia 16 veces inferior a la original y 3 iteraciones de normalización)*[46].



A. 5 regiones más salientes **B.** Mapa de saliencia

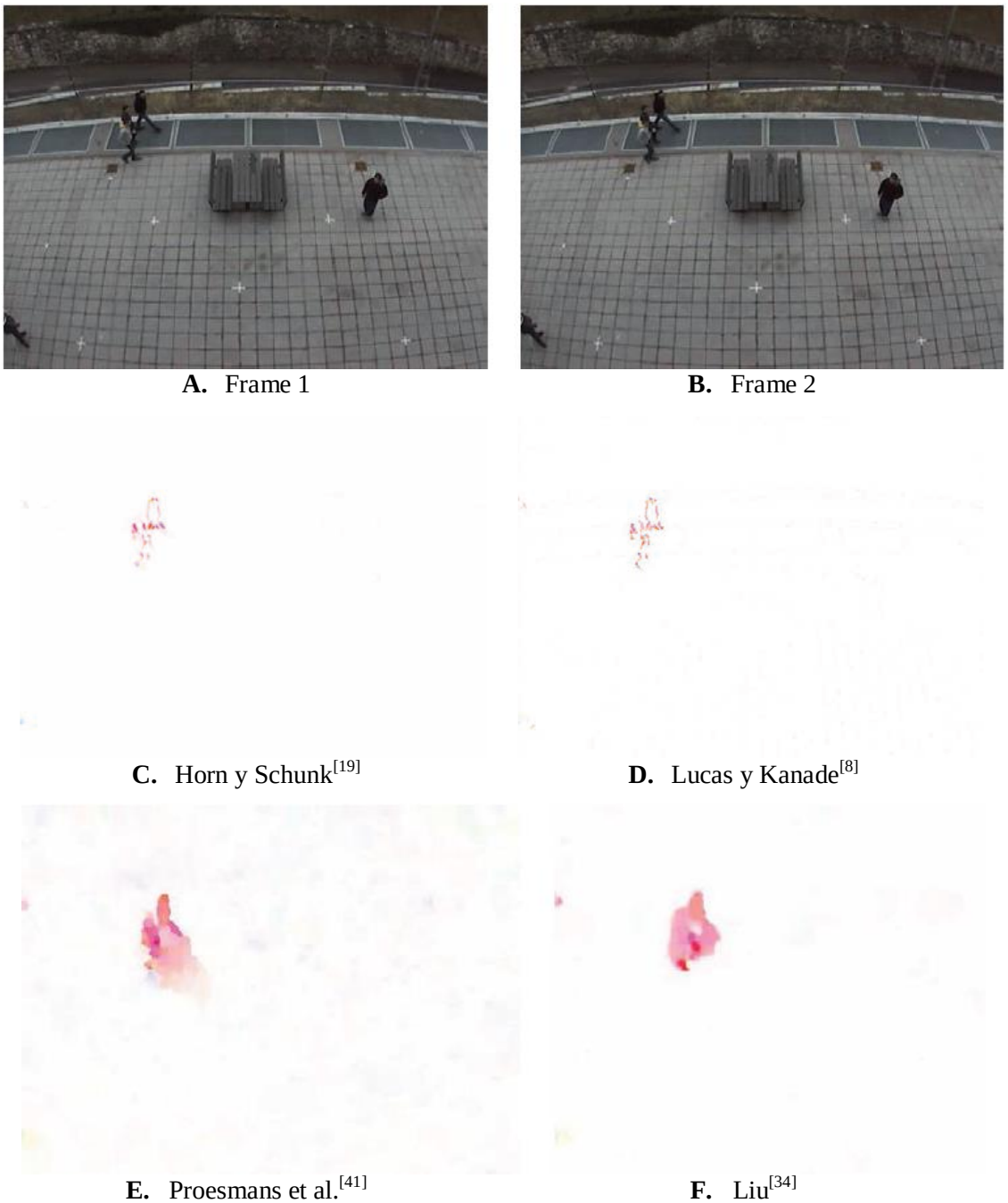
Figura 9. *Resultados de la SaliencyToolbox con los parámetros finales (resolución del mapa de saliencia igual a la de la imagen original y 1 iteración de normalización)*[46].

3.1.2 Cálculo del flujo óptico

Como se ha explicado en la sección 2.1.2, para llevar a cabo un algoritmo de detección de saliencia dinámica es muy común el cálculo del flujo óptico. Esto se debe a que la vista tiende a fijarse en objetos en movimiento. Sin embargo, el cerebro estima la trayectoria seguida por éstos y la vista comienza a buscar otros elementos salientes de la

escena. Por ello, Carlos Ruiz extrajo tanto la velocidad como la aceleración de cada *frame*, para lo que seleccionó el método desarrollado por Liu[34].

Se barajaron cuatro algoritmos de estimación de movimiento en Matlab: el de Horn y Schunk[19]; el de Lucas y Kanade[8]; el de Proesmans et al.[41], y el elegido, desarrollado por Liu[34]. Los cuatro se probaron con diferentes vídeos. Se obtuvo que los dos primeros no detectan correctamente el movimiento en las áreas sin textura, mientras que los otros dos dan lugar a unos resultados parecidos entre sí, siendo el de Liu[34] más efectivo a la hora de estimar pequeños movimientos, y menos ruidoso que el de Proesmans et al.[41] (véase Figura 10).



A. Frame 1 **B.** Frame 2

C. Horn y Schunk[19] **D.** Lucas y Kanade[8]

E. Proesmans et al.[41] **F.** Liu[34]

Figura 10. *Resultados de diversos algoritmos para el cálculo del flujo óptico[46]. En (C) y (D) solamente se detecta el movimiento en el borde de los objetos. (E) y (F) proporcionan resultados parecidos, pero (E) introduce un mayor ruido.*

Empleando el modelo de Liu para el cálculo del flujo óptico, se extrae la velocidad de todos los píxeles entre dos fotogramas consecutivos. A partir de ésta, se calcula la aceleración entre tres *frames*, simulando así el sistema de atención visual. De ambas magnitudes se extrae el módulo, ya que la dirección no es relevante para la determinación de la saliencia.

3.1.3 Segmentación de la imagen

En el TFG de Carlos Ruiz[46] se introdujo un segmentador de imagen por dos razones: la primera es la inexactitud tanto del algoritmo de saliencia estática como de los cálculos del flujo óptico. De esta manera, todos los píxeles de un segmento poseen los mismos descriptores, disminuyendo el problema de oclusiones que se producía al calcular la aceleración. En segundo lugar, se introdujo para compensar la poca precisión del *eye-tracker* con el que se obtuvo el *ground truth*. Así, se clasifica como saliente todo el objeto al que esté mirando el observador. Esto se explica con más detalle en la sección 4.2.

Esta parte se llevó a cabo mediante un software de segmentación llamado EDISON[5], elegido por su sencillez e implementación en Matlab. Este sistema está formado por dos módulos, uno de detección de bordes y otro que se encarga de segmentar la imagen.

3.2 Modificaciones del modelo

Con el objetivo de mejorar las prestaciones del modelo de saliencia dinámica propuesto en el TFG de Carlos Ruiz[46], se han realizado una serie de modificaciones del *software*. La primera que se llevó a cabo, fue la adaptación de éste a la extracción de los datos del *eye-tracker* a partir de matrices en Matlab. Esto se debe a que originalmente se extraían los datos de un fichero *.xml*, en lugar de desde un *.mat*. De esta manera, la extracción de los datos del *eye-tracking* es mucho más sencilla.

En esta sección se van a explicar todos los elementos que se han añadido al modelo de saliencia espacio-temporal, expuesto en el apartado 3.1. En primer lugar se encuentra la introducción de la detección facial, de la que se justificará el modelo empleado y se explicará su implementación en el *software*, proporcionando algunos de los resultados obtenidos. Después, se va a explicar cómo se ha llevado a cabo el descriptor de *center-bias*, que caracteriza la tendencia de las personas a mirar al centro de las imágenes.

3.2.1 Extracción del descriptor facial

3.2.1.1 Elección del algoritmo de detección facial

En el estado del arte se han visto numerosos modelos de detección facial, siendo el de Viola y Jones[55] el más famoso y maduro. Para llevar a cabo esta elección se han establecido una serie de requisitos, como se expone a continuación.

[5] Software EDISON: http://coewww.rutgers.edu/riul/research/code/EDISON/doc/overview.html

En primer lugar, el algoritmo de detección facial debe estar implementado en el mismo lenguaje que el *software* de saliencia dinámica, programado en Matlab. En segundo lugar, se busca un método sencillo, ya que el objetivo perseguido es comprobar que añadiendo este descriptror, mejorará la detección de saliencia en escenas donde aparezcan personas, dejando como trabajo futuro la mejora de las prestaciones del detector facial.

De este modo, para llevar a cabo la búsqueda de rostros se ha empleado la *Computer Vision System Toolbox*[6] de Matlab, que incluye el sistema *vision.CascadeObjectDetector* basado en el algoritmo de Viola y Jones[55], analizado en la sección 2.2.1.1. Además de ser sencillo y tener bajo coste computacional, permite la detección de caras en posición frontal y de perfil. También puede detectar las zonas más relevantes de la cara, como pueden ser los ojos o la boca. Por otro lado, al formar parte de una *toolbox* de Matlab, el sistema está muy bien documentado, lo que facilita la resolución de problemas o dudas a la hora de emplearlo.

El detector de objetos en cascada posee una propiedad llamada *ClassificationModel*, con la que se puede seleccionar qué se quiere detectar: una cara en posición frontal, de perfil, los ojos, la nariz o la boca. Se realizaron diferentes pruebas con el fin de seleccionar el mejor modo de detectar todas las caras de un *frame*, y se llevó a cabo una combinación de detección facial frontal y de perfil.

A. Detección de caras de perfil. **B.** Detección de caras de frente.

C. Detección de boca **D.** Detección de ojo izquierdo

Figura 11. *Resultados para diferentes modelos de clasificación de vision.CascadeObjectDetector.*

[6] *Computer Vision System Toolbox*: http://es.mathworks.com/products/computer-vision/

Como se puede ver en la figura anterior, si se realiza una combinación entre detección de perfil y frontal, se puede llevar a cabo la localización todas las caras. Sin embargo, a la hora de detectar otras partes del rostro se producen errores como en (C) y (D). Además, otros rasgos como son los ojos o la nariz ni siquiera son detectados.

3.2.1.2 Implementación del descriptor

En primer lugar se emplea el sistema *vision.CascadeObjectDetector*[7] para detectar la localización de las caras en un *frame* del vídeo a analizar. El detector de objetos en cascada emplea el algoritmo de detección de Viola y Jones[55], además de un modelo de clasificación que por defecto está configurado para detectar caras, pero se puede usar para la detección de otros objetos.

Una opción barajada inicialmente fue realizar el seguimiento de los rostros durante el vídeo, para lo que se utilizaría el algoritmo de Kanade-Lucas-Tomasi (también conocido como KLT)[8], implementado en la misma *toolbox* de Matlab que el detector facial. Tras implementarlo y evaluar sus prestaciones, se obtuvo que el coste computacional era similar al obtenido sin este seguimiento, debido a que la mayor carga computacional se produce a la hora de calcular el resto de descriptores. Además, era necesaria la introducción de un detector de cambio de escena mientras que sin seguimiento facial esto no era preciso, ya que en cada *frame* se lleva a cabo una detección.

Para evaluar el coste computacional del modelo con seguimiento y sin seguimiento facial de manera informal, se llevó a cabo el cálculo de la saliencia dinámica para ambas situaciones con las mismas condiciones: el mismo vídeo, la misma red neuronal y el mismo equipo, dedicado a esta única tarea. Se obtuvo que el tiempo empleado para el caso en el que hay seguimiento, el vídeo se tarda en procesar 34 minutos, mientras que sin seguimiento tarda 37. De esta manera, se concluye que la diferencia de tiempo es insignificante para la dificultad que conllevaría introducir el seguimiento facial, debido a la necesidad de un detector de cambios de escena, en el caso de que se evalúen vídeos con varias escenas.

Cuando se ha finalizado el análisis de un *frame* en busca de rostros, se dibuja una forma geométrica rectangular rellena sobre estos, con la función *insertShape*, y se calcula una imagen binaria, en la que los píxeles pertenecientes a las caras tienen valor 1 y el resto 0.

El umbral del detector de objetos en cascada tiene como valor por defecto 4 aunque es modificable, por lo que se han realizado distintas pruebas –de manera informal– para encontrar el valor que de lugar a unas mejores prestaciones. En la siguiente tabla se pueden ver los valores aproximados obtenidos para distintos vídeos y distintos valores del umbral. Los vídeos están extraídos de la base de datos del *eye-tracker*, que se explica con más detalle en el capítulo de experimentos, y sólo se muestran los resultados de algunos, aunque todos han sido evaluados.

[7] *Vision.CascadeObjectDetector*: http://es.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html
[8] KLT: http://es.mathworks.com/help/vision/examples/face-detection-and-tracking-using-the-klt-algorithm.html

	Clip 46			**Clip 50**			**Clip 55**			**Clip 60**		
Umbral	9	7	5	9	7	5	9	7	5	9	7	5
Porcentaje caras detectadas	61,8%	74%	91,8%	84,2%	92,2%	100,8%	29,4%	41,6%	58%	78,7%	89,3%	100,65%

Tabla 2. *Prestaciones del detector facial para distintos valores del umbral.*

El porcentaje de caras detectadas se ha calculado siguiendo la ecuación 3.1:

$$Caras\ detectadas\ (\%) = \frac{\sum_{frame} \#\ caras\ detectadas}{\sum_{frame} \#\ caras} \cdot 100 \qquad (3.1)$$

Es decir, que si en el vídeo hay 100 fotogramas y aparecen dos personas, en cada *frame* debe haber 2 caras, siendo el denominador de la ecuación anterior igual a 200.

En la Tabla 2, se observa que los valores más altos de caras detectadas se dan para un umbral igual a 5. Esto se debe a la detección de falsos positivos, por eso en algunos casos el número de caras detectadas es mayor al número real de éstas.

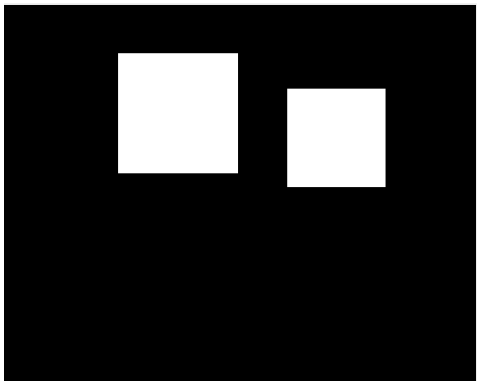
Sin embargo, para un umbral igual a 9, las prestaciones se reducen considerablemente. De este modo, el valor elegido para el umbral es 7, ya que no se desea desechar ninguna cara (falsos negativos), pero tampoco clasificar como rostros elementos que no lo son (falsos positivos).

Por último, cabe mencionar que este umbral no será óptimo para todos los vídeos, ya que depende de las condiciones de iluminación, resolución, etc. Se ha aplicado el mismo valor del umbral para la detección frontal y de perfil de las caras.

3.2.1.3 Resultados

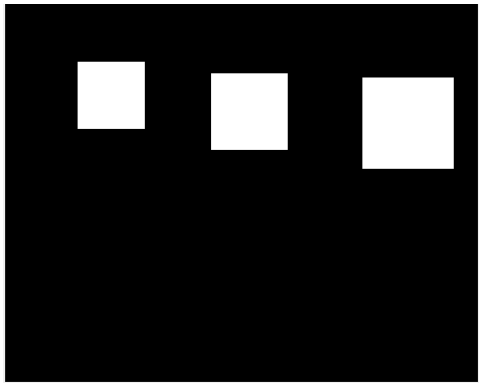
A continuación, se van a mostrar una serie de resultados del descriptor facial programado, constrastando los diferentes umbrales anteriormente mencionados. Se ha elegido aleatoriamente el *frame* 50 de algunos de los vídeos de la base de datos, para mostrar el descriptor facial.

A. *Clip 46*, fotograma 50.



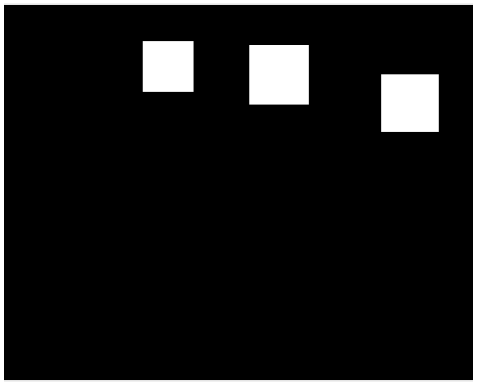
B. Descriptor de caras. Umbral = 7.

C. *Clip 47*, fotograma 50.



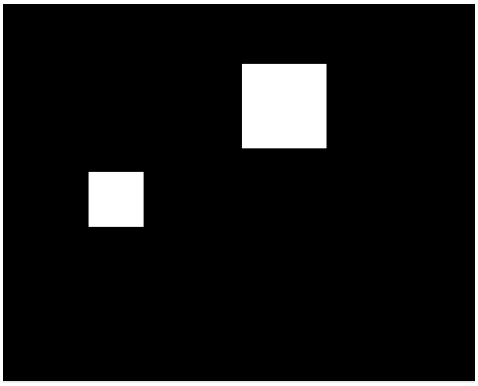
D. Descriptor de caras. Umbral = 7.

E. *Clip 56*, fotograma 50.



F. Descriptor de caras. Umbral = 7.

G. *Clip 60*, fotograma 50.



H. Descriptor de caras. Umbral = 7.

Figura 12. *Extracción del descriptor facial para un umbral igual a 7. En (B), (D), (F) y (H) se puede ver que la región detectada como un rostro tiene valor 1 (color blanco) y el resto 0 (color negro).*

En el caso de disminuir el umbral a 5, se producen falsos positivos. Si, por ejemplo, observamos los resultados del *frame* 50 perteneciente al *clip 60* para este valor, obtenemos el descriptor de la Figura *13*.

A. *Clip 60*, fotograma 50. **B.** Descriptor de caras. Umbral = 5.

Figura 13. *Extracción del descriptor facial para un umbral igual a 5. En (B) se observa un falso positivo.*

Sin embargo, si aumentamos el umbral a 9, hay ocasiones en las que se producen falsos negativos como se ve en la siguiente Figura 14.

A. Clip 46, frame 50. **B.** Descriptor de caras. Umbral = 9.

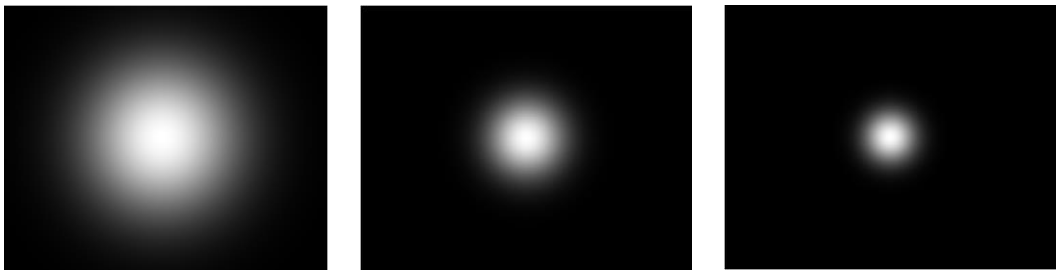
Figura 14. *Extracción del descriptor facial para un umbral igual a 9. En (B) se observa un falso negativo.*

3.2.2 Extracción del descriptor de *center-bias*

El hecho de que el centro de una imagen es la región que primero observan las personas se ha demostrado en numerosos experimentos[2]. Por esta razón, se ha decidido introducir un descriptor que aporte mayor importancia a esta región. En el capítulo 4 se comprobará si da lugar a un mejor o peor funcionamiento del algoritmo de saliencia espacio-temporal.

Esto se ha llevado a cabo mediante la programación en Matlab de una matriz del mismo tamaño que los fotogramas del vídeo analizado, perteneciendo cada valor de ésta a un píxel del *frame*. Esta matriz, inicializada a 0, tiene como valor una gaussiana en el centro de la imagen.

Para cubrir los casos en los que el *zoom* de la escena sea menor o mayor, se han calculado tres matrices con una gaussiana de desviación típica diferente (5, 10 y 15 veces más pequeña al lado menor del fotograma), dando lugar a distintos radios, como se ve en la Figura 15. De esta manera, se obtienen tres descriptores de *center-bias*.



A. σ = lado menor/5 **B.** σ = lado menor/10 **C.** σ = lado menor/15

Figura 15. *Descriptores de center-bias para los tres valores de desviación típica.*

Al entrenar la red neuronal con los tres descriptores, éstos se superponen haciendo tres veces mayor la probabilidad de que la región de radio menor sea saliente. Entre el radio menor y el mediano, la probabilidad es el doble, y entre el mediano y el grande es igual a 1.

3.2.3 Entrenamiento de la red neuronal

Para llevar a cabo el entrenamiento de la red neuronal se ha hecho uso de una *toolbox* de Matlab, llamada *Neural Network Toolbox*[9], que ha sido elegida por su sencillez y flexibilidad. Permite llevar a cabo el entrenamiento desde su interfaz gráfica de una manera rápida y sencilla, ya que solo hay que seleccionar el valor para algunos parámetros, como la cantidad de neuronas en la capa oculta.

En primer lugar, hay que elegir el tipo de problema para el que se entrena la red neuronal, en este caso queremos clasificar las regiones de una imagen como salientes o no salientes (1 ó 0), por lo que elegimos *Pattern recognition and classification* (véase Figura 16). De esta manera, se emplea una red neuronal de tipo perceptrón multicapa.

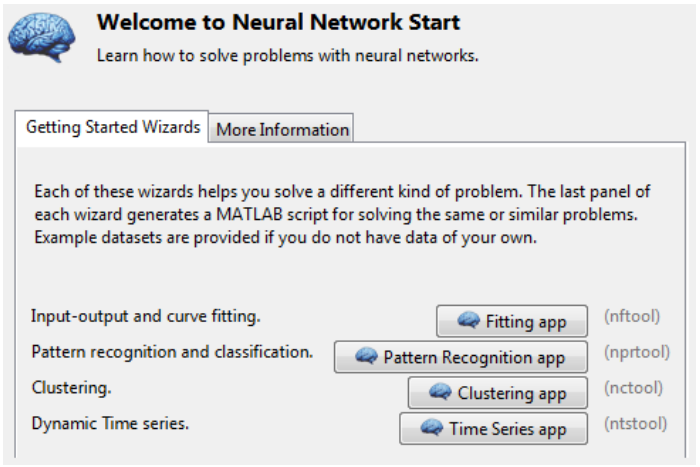


Figura 16. *Pantalla inicial de la Neural Network Toolbox de Matlab.*

[9] *Neural Network Toolbox*: http://es.mathworks.com/products/neural-network/

Después, hay que introducir la matriz de entradas y objetivos de la red neuronal y elegir la proporción de datos empleados para entrenar, validar y comprobar las prestaciones de la red. Por último, se selecciona el número deseado de neuronas en la capa oculta y se entrena.

Una vez se ha entrenado la red neuronal, se pueden analizar los resultados mediante diferentes gráficos como la curva ROC o la matriz de confusión. Si se necesitan modificar más parámetros, al final del proceso de entrenamiento la *toolbox* te permite acceder al código. En nuestro caso, se han fijado los datos que se van a emplear como grupo de entrenamiento, validación y prueba (en inglés, *train*, *validation* y *test*, respectivamente).

Capítulo 4

Experimentos

Una vez implementados los nuevos descriptores, la siguiente fase es llevar a cabo el entrenamiento de la red neuronal. Para esto, se van a emplear 8 vídeos (sección 4.1) con información de las fijaciones oculares. De este modo, se crean tres conjuntos: uno de entrenamiento, otro de validación y otro de evaluación. Cada conjunto contiene unas entradas para la red neuronal, que serán los descriptores extraídos para los diferentes vídeos, y unas salidas binarias, que determinan la saliencia.

Todo esto será visto en detalle a lo largo del capítulo, en el que primero se especificará la base de datos de la que se han extraído los vídeos, después se explicará cómo se ha llevado a cabo el entrenamiento de la red neuronal y por último, se analizarán los resultados del modelo de saliencia visual propuesto.

4.1 Base de datos de fijaciones oculares

La base de datos empleada cuenta con 60 secuencias de vídeo que poseen información extraída por un *eye-tracker*. Ésta fue desarrollada por Coutrot y Guyader, con el fin de estudiar cómo influye el sonido en los movimientos oculares a la hora de visualizar un vídeo[12]. Por tanto, las fijaciones de los observadores van a depender del audio. Cabe mencionar que en este trabajo solamente se han tenido en cuenta las características visuales y no las auditivas dejando para futuros trabajos el análisis de la modalidad auditiva. A continuación se van a enumerar los elementos que Coutrot y Guyader emplearon para crear la base de datos:

a) Vídeos: organizados en cuatro categorías con 15 vídeos cada una: un objeto en movimiento (clips 1 a 15), varios objetos en movimiento (clips 16 a 30), paisajes (clips 31 a 45) y caras (clips 46 a 60). Tienen una resolución de 720x576 píxeles y un ratio de fotogramas por segundo igual a 25. Duran entre 10 y 25 segundos y constan de una sola escena.

b) Participantes: 20 estudiantes de la Universidad de Grenoble, con visión y audición normal, que no conocían el propósito del experimento.

c) Eye-tracker: el equipo empleado fue el Eyelink 1000 de SR Research. La frecuencia de muestreo de este aparato es de 1000 Hz, por lo que los resultados serán muy aproximados a la realidad.

d) Pantalla: para llevar a cabo el experimento, se dispuso a los participantes a 57 cm de una pantalla CRT de 21 pulgadas, con una resolución de 1024x768 píxeles y una tasa de refresco de 75 Hz.

Los datos recogidos por el *eye-tracker* están organizados en cinco conjuntos, y cada uno contiene las fijaciones para los 60 vídeos con diferentes condiciones sonoras. Puesto que el objetivo de los autores era estudiar cómo influye el sonido en la atención visual, modificaron las pistas de audio de los vídeos para dar lugar a cuatro de estos conjuntos. El restante, que posee el sonido original, es el que ha sido elegido para extraer el *ground truth* necesario para entrenar la red neuronal, de manera que las condiciones sonoras influyan lo menos posible en los resultados.

A continuación se van a especificar los vídeos empleados para llevar a cabo los experimentos, justificando su elección.

En su TFG, Carlos Ruiz encontró grandes problemas a la hora de entrenar el modelo de saliencia dinámica, debido a que los datos de las fijaciones oculares de los que disponía, habían sido recogidos por un *eye-tracker* de precisión y frecuencia de muestreo bajas. De este modo, entrenó el sistema con un solo vídeo.

Sin embargo, esta vez se dispone de una base de datos robusta, en cuya extracción participaron 20 individuos (de ahora en adelante, participantes), que consta de 60 vídeos. Aunque lo óptimo sería entrenar la red neuronal con el máximo número posible de vídeos, el tiempo de procesamiento de uno de ellos con el fin de extraer los descriptores y el *ground truth* de todos los participantes, es demasiado elevado (aproximadamente 2 horas por cada 20 segundos de vídeo, con una resolución de 360x288 píxeles). De esta manera, se ha realizado una selección informal siguiendo los criteros descritos a continuación.

Se han escogido los vídeos considerados más adecuados para obtener unos buenos resultados de entrenamiento. Para esto, primero se han descartado los vídeos de paisajes debido a su escaso movimiento. También se ha prescindido de los que presentaban movimiento de cámara, ya que el cálculo del flujo óptico –explicado en la sección 3.1.2– que lleva a cabo el algoritmo de saliencia dinámica, no lo compensa. De esta manera, clasificaría como salientes regiones que no lo son solamente por este movimiento.

Por otro lado, de los vídeos que contienen rostros, se han elegido los que mejores resultados proporcionan a la hora de la detección facial, ya que nuestro objetivo es

comprobar si la introducción de esta característica mejora los resultados del modelo de saliencia dinámica, dejando como futuro trabajo la mejora de las prestaciones del detector.

Así, para el entrenamiento y validación de la red neuronal se han empleado los clips 46, 47 y 60, del grupo de vídeos que contienen caras, y 6, 15 y 17 del resto de categorías. Para el conjunto de prueba se han escogido los vídeos 28 y 50. Ninguno de éstos, por tanto, presenta movimiento de cámara, ni pertenecen a la categoría de paisajes.

4.2 Obtención del *ground truth*

Para obtener el *ground truth* se emplean 8 vídeos (6 de entrenamiento y validación y 2 de *test*), como se ha indicado anteriormente, con datos de las fijaciones oculares de múltiples participantes. Los datos contienen las coordenadas de cada *frame* a las que éstos miraron, por lo que cada imagen posee más de una fijación –una por cada observador–. Así, es necesario desarrollar un método para determinar el *ground truth* del conjunto de todos los participantes para cada fotograma. A continuación se explica cómo se ha llevado a cabo esta tarea para un solo vídeo. Para calcular el *ground truth* de todos los vídeos, basta con concatenar las matrices obtenidas para cada uno.

En primer lugar, se realiza un procesado de cada *frame* que incluye tres etapas: obtener los mapas de características (saliencia espacial, velocidad, aceleración, caras y *center-bias*), segmentar la imagen y leer los datos del *eye-tracker* pertenecientes a ese *frame*. Después, se calcula la media de estas características para cada segmento, que serán las entradas de la red neuronal.

En segundo lugar, se determina a qué segmento está mirando cada participante. Como nuestros datos solo incluyen las coordenadas de un píxel, se establece una circunferencia de radio configurable (R), de manera que todos los segmentos que poseen algún píxel en su interior, se establecen como salientes (valor 1). El resto se establecen como no salientes (valor 0). Estos valores serán la salida de la red neuronal. Para este radio se ha elegido un valor de 100 píxeles, de manera que la circunferencia se ajustara correctamente a los vídeos empleados.

De esta forma, se guardan los valores de las entradas y las salidas para cada participante. Sin embargo, los primeros serán comunes para todos, ya que el procesamiento del vídeo es el mismo, lo que difiere para cada uno son los segundos valores. Por tanto, se ha decidido calcular una matriz de salidas común, para lo que se van a proponer tres aproximaciones:

a) Suma: se calcula la suma de las salidas para todos los participantes y se binariza el resultado. De esta manera, cada *frame* tiene más de una región saliente –una por cada observador–.

b) Media: se calcula la suma de las salidas para todos los participantes y se divide entre la cantidad de éstos. El resultado se redondea para binarizarlo.

Así, solamente son salientes los segmentos de un *frame* que han sido observados por más de la mitad de los participantes.

c) Moda: como en los anteriores casos, se calcula la suma y se determinan salientes las regiones que han sido observadas por más del 75% de los participantes.

Si no se llevase a cabo la creación de una matriz de salidas común, la cantidad de datos para entrenar la red neuronal sería muy grande, ya que habría que incluir todas las entradas de todos los participantes –que son iguales para todos–. Esto supondría un enorme tiempo de entrenamiento, por lo que eliminando datos redundantes, hacemos que éste se reduzca.

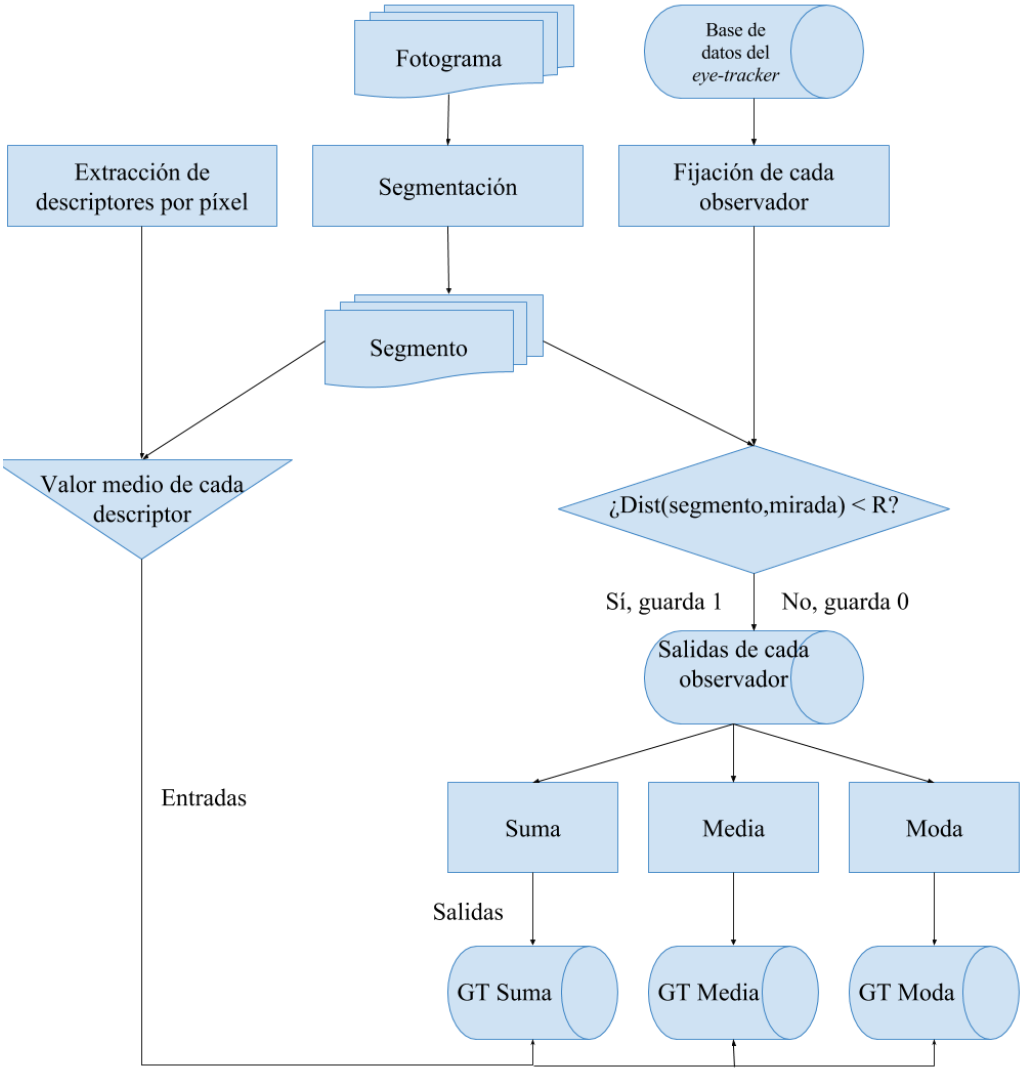


Figura 17. *Diagrama de extracción de los diferentes ground truth.*

Además, se quiere evaluar las prestaciones obtenidas al usar por un lado los mapas de características obtenidos por Carlos Ruiz (saliencia, velocidad y aceleración), y por otro evaluar la introducción de los nuevos mapas, por lo que se obtienen nueve bases de datos (véase Tabla 3).

Nombre	Mapas de características	Positivos	Negativos	Segmentos totales
GT_Ruiz_Suma	Saliencia estática, velocidad y aceleración	44,2%	55,8%	601.395
GT_Ruiz_Media		11%	89%	
GT_Ruiz_Moda		4,5%	95,5%	
GT_Caras_Suma	Saliencia estática, velocidad, aceleración y caras	44,2%	55,8%	
GT_Caras_Media		11%	89%	
GT_Caras_Moda		4,5%	95,5%	
GT_CenterBias_Suma	Saliencia estática, velocidad, aceleración, caras y *center-bias*	44,2%	55,8%	
GT_CenterBias_Media		11%	89%	
GT_CenterBias_Moda		4,5%	95,5%	

Tabla 3. *Bases de datos generadas para entrenamiento y validación de la red neuronal y su composición.*

Para el conjunto de prueba, extraído de los vídeos 28 y 50, se ha realizado el mismo procedimiento, obteniendo bases de datos con las mismas características que las de la Tabla 3, aunque con diferente tamaño y proporción de positivos y negativos.

No obstante, al entrenar la red neuronal con estos datos, en algunos casos se produce una gran cantidad tanto de falsos positivos, como de falsos negativos. Además, algunas de estas bases de datos están muy desequilibradas, en cuanto a porcentaje de positivos y negativos. Estos resultados se estudian con más detalle en la sección 4.3, donde se concreta cómo se ha llevado a cabo el entrenamiento de la red neuronal y se muestran las prestaciones obtenidas para cada conjunto.

El motivo de estos malos resultados, son los numerosos *outliers* incluidos a la hora de establecer una región saliente. Si recapitulamos, en esta misma sección se establece que todos los segmentos que tengan algún píxel en el interior del círculo de radio R, serán salientes. Sin embargo, en la Figura 18 se muestra cómo es posible que se incluyan segmentos con una mayoría de píxeles fuera del radio. Por este motivo, se han desarrollado otras bases de datos en las que se eliminan estos *outliers*.

A. Imagen original **B.** Imagen de saliencia.

Figura 18. *Región saliente con outliers. (A) es un frame original del clip 60. En (B) se puede la región determinada como saliente (en verde).*

A este problema también se enfrentó Carlos Ruiz en su TFG, por lo que se va a llevar a cabo una solución similar a la que él empleó, expuesta a continuación.

Cuando se produce una fijación ocular en una zona de la imagen, se debe a que alguno de los descriptores destaca sobre el resto[46], de manera que se pueden establecer como salientes las regiones que tienen un valor alto para alguno de ellos. Así, se ha determinado que un segmento será saliente, cuando uno o más de sus descriptores estén por encima del 60% del valor máximo alcanzado por ese descriptor, en alguno de los segmentos del *frame*.

De esta manera se dan cuatro posibilidades ilustradas en la Tabla 4. Los *outliers* son descartados, almacenando en la base de datos con la que se entrenará la red neuronal, únicamente los segmentos que con más seguridad serán salientes o no –positivos y negativos–.

	Dist(segmento, mirada)<R	**Característica > 60% de su máximo en un *frame***
Positivo	Sí	Sí
Outlier positivo	Sí	No
Negativo	No	No
Outlier negativo	No	Sí

Tabla 4. *Procedimiento de elección de outliers.*

Con este procedimiento, obtenemos una matriz de entradas diferente para cada participante, ya que los segmentos son clasificados como *outliers* dependiendo de dónde hayan fijado su mirada. De este modo, se van a concatenar los datos de todos los participantes, tanto las entradas como las salidas, sin llevar a cabo una matriz conjunta como en el caso anterior.

Sin embargo, la base de datos generada posee más de siete millones de entradas y tiene un 5% de segmentos salientes –positivos– y un 95% de segmentos no salientes –negativos–. Esta gran cantidad de datos supone un excesivo tiempo de entrenamiento de la red y, además, el número de positivos y negativos está muy desequilibrado. Por estos motivos, se ha optado por diezmar las muestras negativas, obteniendo las bases de datos indicadas en la Tabla 5. En este caso, el número de segmentos es diferente para cada una, ya que dependiendo de los descriptores empleados, se producen unos *outliers* u otros.

Nombre	Mapas de características	Positivos	Negativos	Segmentos totales
GT_Ruiz_2575	Saliencia estática, velocidad y aceleración	25%	75%	604.016
GT_Ruiz_5050		50%	50%	302.008
GT_Caras_2575	Saliencia estática, velocidad, aceleración y caras	25%	75%	1.078.924
GT_Caras_5050		50%	50%	539.462
GT_CenterBias_2575	Saliencia estática, velocidad, aceleración, caras y *center-bias*	25%	75%	1.810.888
GT_CenterBias_5050		50%	50%	905.444

Tabla 5. *Bases de datos generadas para entrenamiento y validación de la red neuronal y su composición.*

Sin embargo, estas últimas bases de datos han sido solamente utilizadas como conjunto de entrenamiento y validación de la red neuronal, ya que el conjunto de prueba no debe estar diezmado. Además, las salidas del test tienen que estar extraídas mediante consenso de todos los participantes, es decir, hay que llevar a cabo una matriz de entradas y salidas conjunta para todos. De lo contrario, se estaría probando la red neuronal con datos contradictorios.

Así, el conjunto de test para estas bases de datos ha sido obtenido con los vídeos 28 y 50, que han sido procesados con el fin de obtener los segmentos salientes y no salientes. Después se han buscado los *outliers* y en lugar de eliminarlos, se ha igualado su saliencia a 0. De esta manera, las matrices de entrada y salida de todos los participantes tienen el mismo número elementos, por lo que se puede llevar a cabo una matriz conjunta. Se han realizado tres tipos de matrices conjuntas: calculando la suma, la media y la moda de las salidas de los participantes, siguiendo el mismo procedimiento para su cálculo que el explicado anteriormente.

Se podrían haber extraído muchas más bases de datos para entrenar la red neuronal, sin embargo, debido al tiempo consumido en su entrenamiento, se ha decidio probar con las anteriormente expuestas. Aunque este trabajo está centrado en comprobar si la incorporación de las dos características de alto nivel (caras y *center-bias*), da lugar a una mejor predicción de la saliencia visual, se ha propuesto, en la sección 5.2, como trabajo futuro llevar a cabo la mejora del entrenamiento de la red.

4.3 Entrenamiento de la red neuronal

En este apartado se van a mostrar los resultados obtenidos al entrenar la red neuronal con las bases de datos explicadas en la sección anterior. Finalmente, se elegirá la red neuronal óptima para evaluar el modelo de saliencia desarrollado en este TFG. Esta evaluación se realiza en el siguiente apartado (sección 4.4).

Para todos los experimentos se ha empleado un perceptrón multicapa de 20 neuronas en la capa oculta. El número de neuronas se ha escogido en base a experimentos preliminares en donde se consideró un compromiso entre los requisitos computacionales y los resultados de la detección.

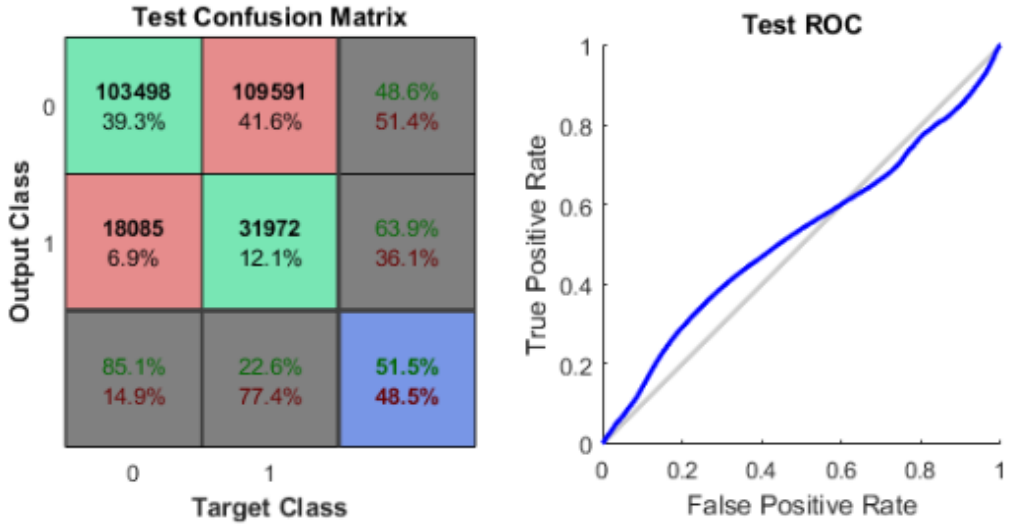
Para evaluar el entrenamiento, se van a obtener dos gráficas: la curva ROC, que muestra la tasa de verdaderos positivos frente a falsos positivos, y la matriz de confusión, que permite ver el porcentaje de segmentos correctamente clasificados –verdaderos positivos y negativos–, así como los falsos positivos y negativos.

En primer lugar, se van a analizar los resultados obtenidos para las primeras bases de datos, construidas mediante el método del diagrama ilustrado en la Figura 17. Como se puede ver en las siguientes figuras, el entrenamiento de la red neuronal no es satisfactorio con ninguna de estas.

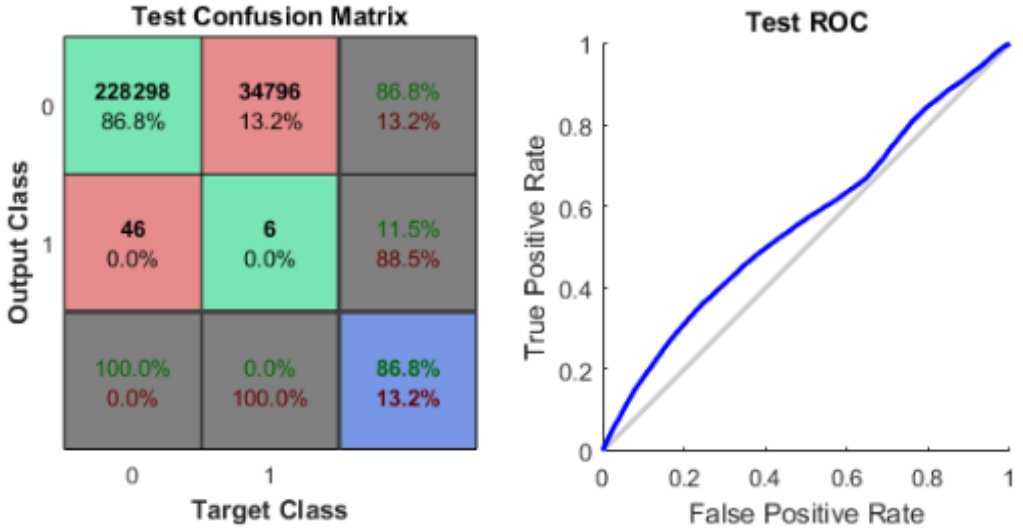
En los casos en los que el consenso entre los participantes, se ha alcanzado mediante la suma de sus matrices de salida es decir, que se tienen en cuenta las regiones salientes de todos los observadores, se producen altos porcentajes de falsos positivos y negativos. Esto es debido a la existencia de *outliers*, como se explicó en la sección 4.2.

Sin embargo, cuando se calcula la media o la moda, la cantidad de entradas negativas es muy superior a la de positivas, por lo que la red neuronal clasifica todas las entradas de test como no salientes.

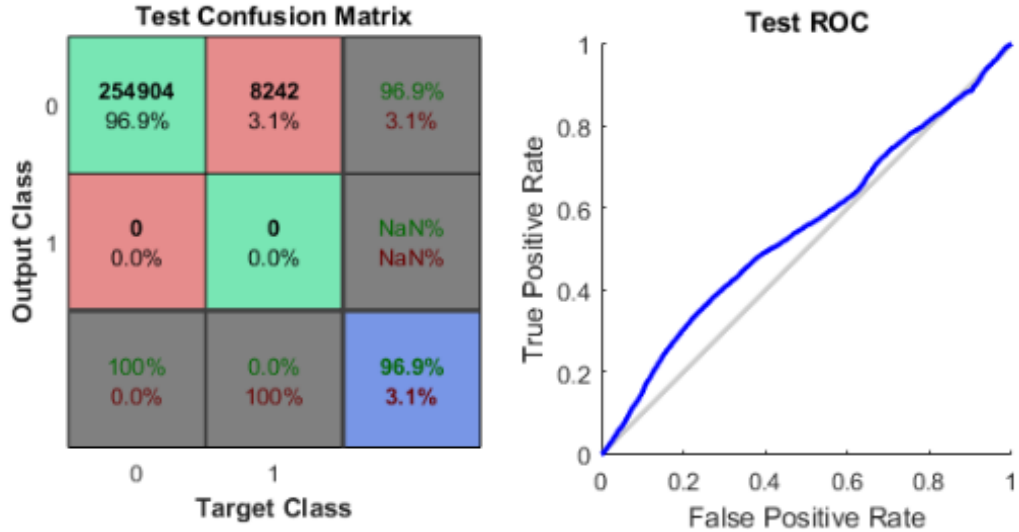
Por otro lado, aunque estos resultados no son muy esclarecedores, si sólo evaluamos las bases de datos obtenidas con la suma de las salidas (Figura 19 (A), Figura 20 (A) y Figura 21 (A)), se puede ver que en el caso de emplear todos los descriptores (Figura 21 (A)), la red neuronal entrenada posee un mayor porcentaje de acierto –73,2% frente a 51,5% y 55,5%–.



A. 'GT_Ruiz_Suma'

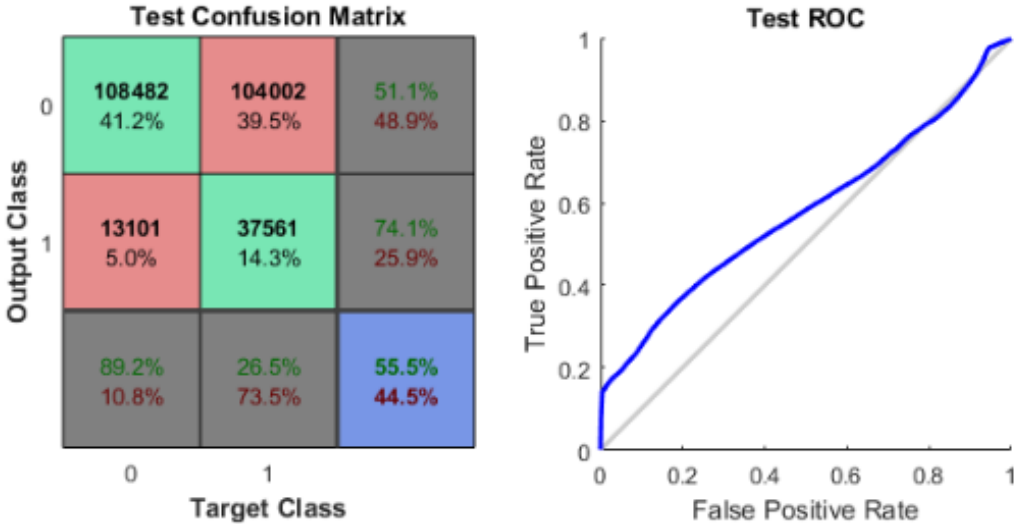


B. 'GT_Ruiz_Media'

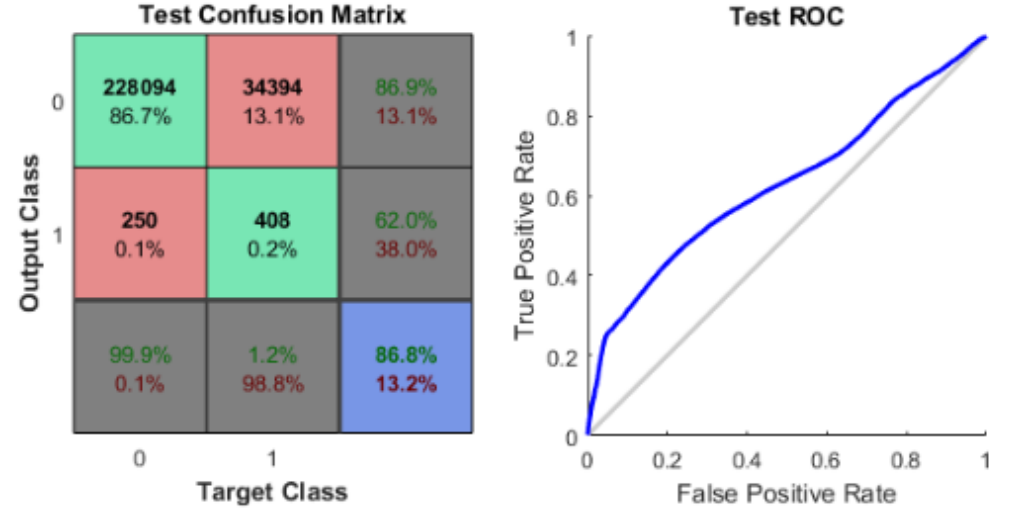


C. 'GT_Ruiz_Moda'

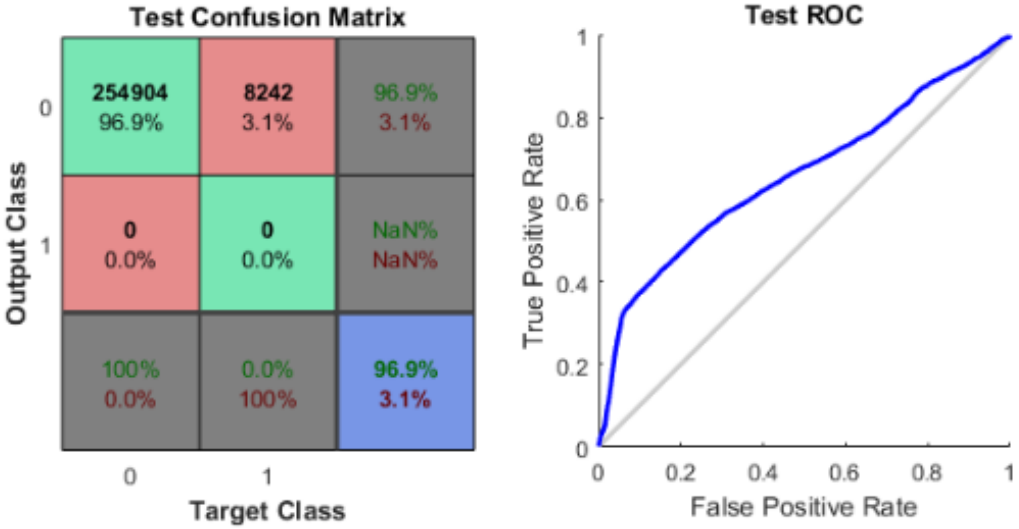
Figura 19. *Resultados del entrenamiento, según la base de datos, usando únicamente las características de saliencia bottom-up.*



A. 'GT_Caras_Suma'

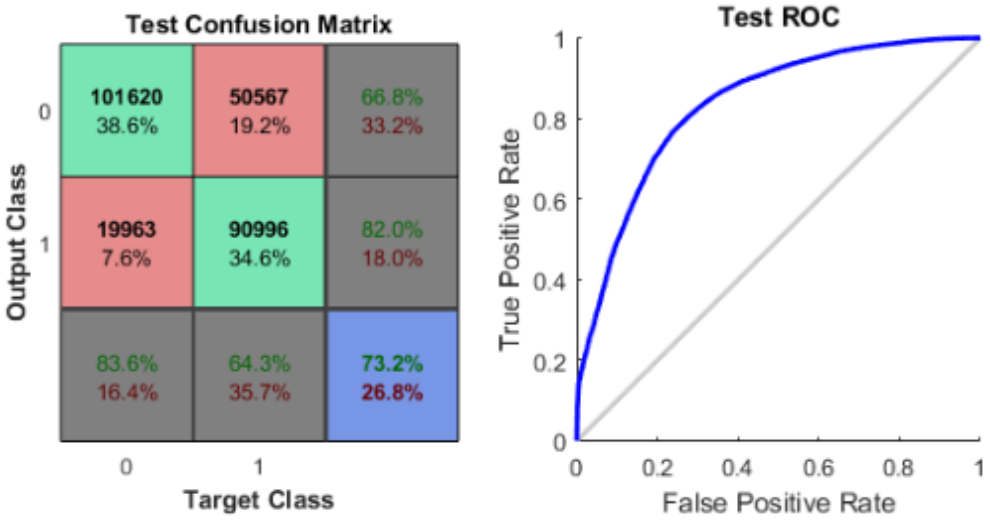


B. 'GT_Caras_Media'

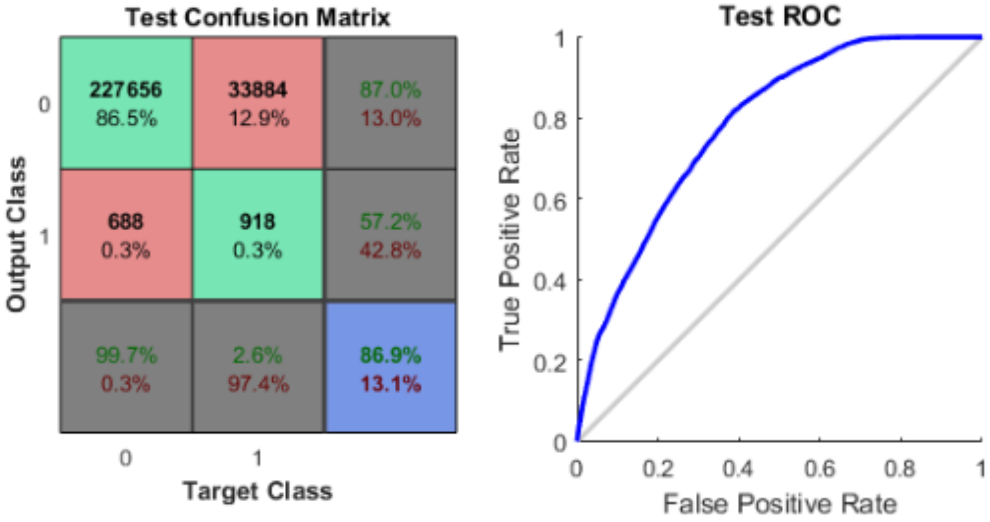


C. 'GT_Caras_Moda'

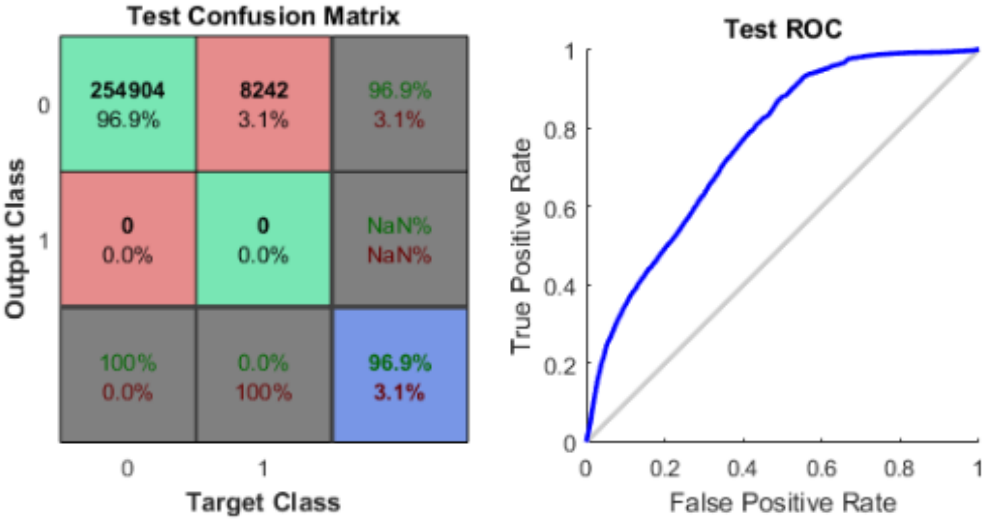
Figura 20. *Resultados del entrenamiento, según la base de datos, usando las características de saliencia bottom-up y el descriptor facial.*



A. 'GT_CenterBias_Suma'



B. 'GT_CenterBias_Media'



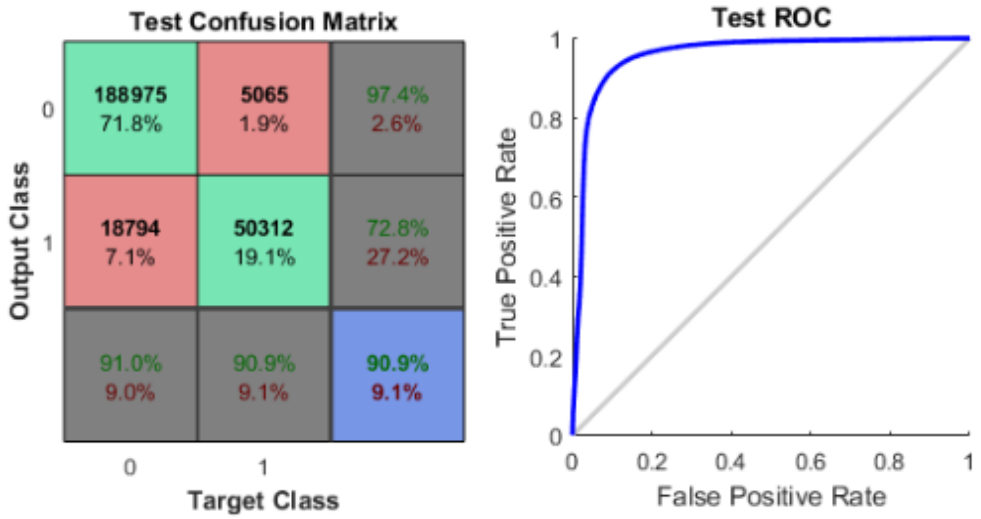
C. 'GT_CenterBias_Moda'

Figura 21. *Resultados del entrenamiento, según la base de datos, usando las características de saliencia bottom-up, el descriptor facial y el de center-bias.*

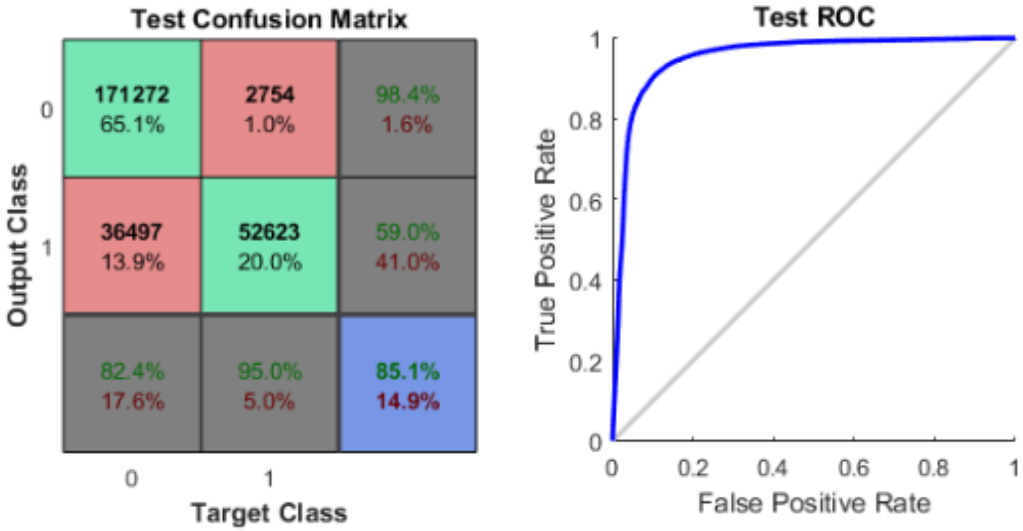
Las prestaciones obtenidas al entrenar la red neuronal con las otras bases de datos, obtenidas al aplicar el umbral del 60% a los descriptores y equilibrar las muestras positivas y negativas, son mucho mejores, como se verá a continuación.

En la sección 4.3 se explicó que las bases de datos de la Tabla 5, son empleadas solamente para el entrenamiento y validación de la red neuronal, mientras que el test se realiza con otro conjunto, en el que sí se ha llevado a cabo una salida consensuada para todos los participantes y en el que se respetan las proporciones naturales de datos positivos y negativos. En la Figura 22 se muestran los resultados obtenidos al utilizar tanto las características *bottom-up* como las *top-down* para los diferentes conjuntos de prueba: suma, media y moda.

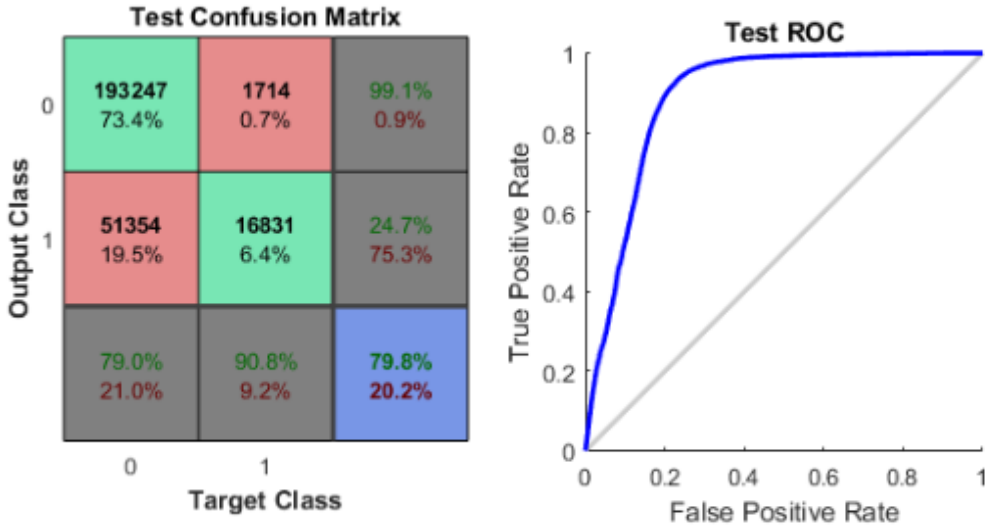
Como en el caso anterior, la suma da mejores resultados que la media o la moda tanto para 'GT_CenterBias_5050' como para 'GT_CenterBias_2575', por lo que de ahora en adelante, se usará el conjunto de test obtenido mediante este procedimiento.



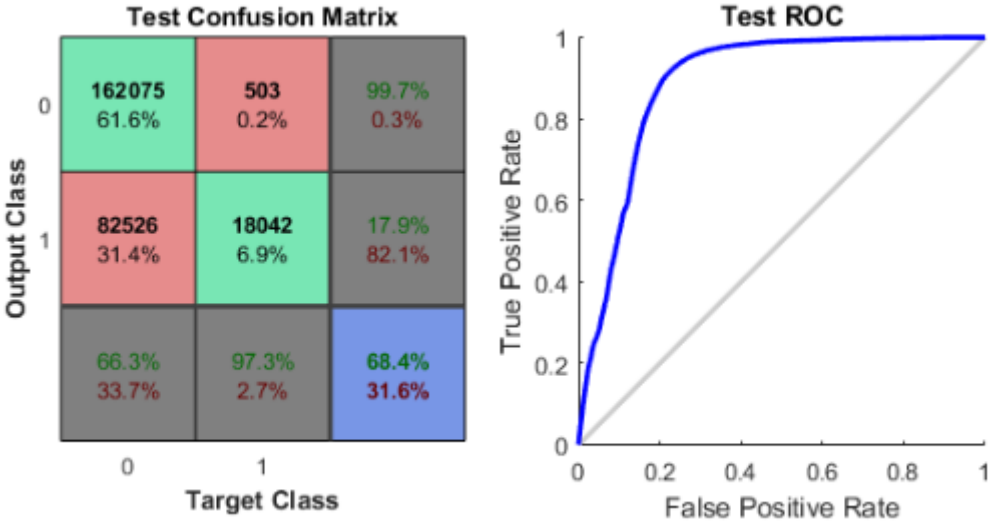
A. Entrenamiento y validación: 'GT_CenterBias_2575'. Test: suma.



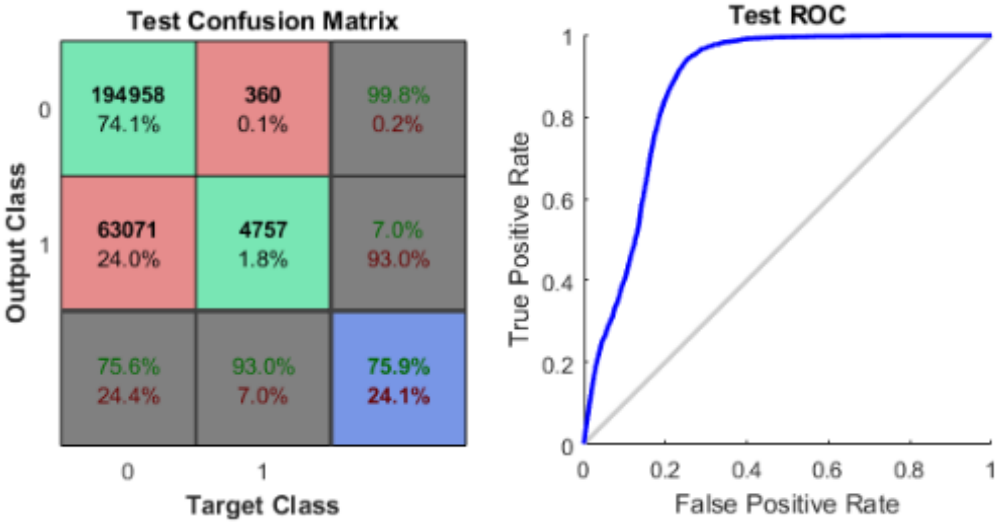
B. Entrenamiento y validación: 'GT_CenterBias_5050'. Test: suma.



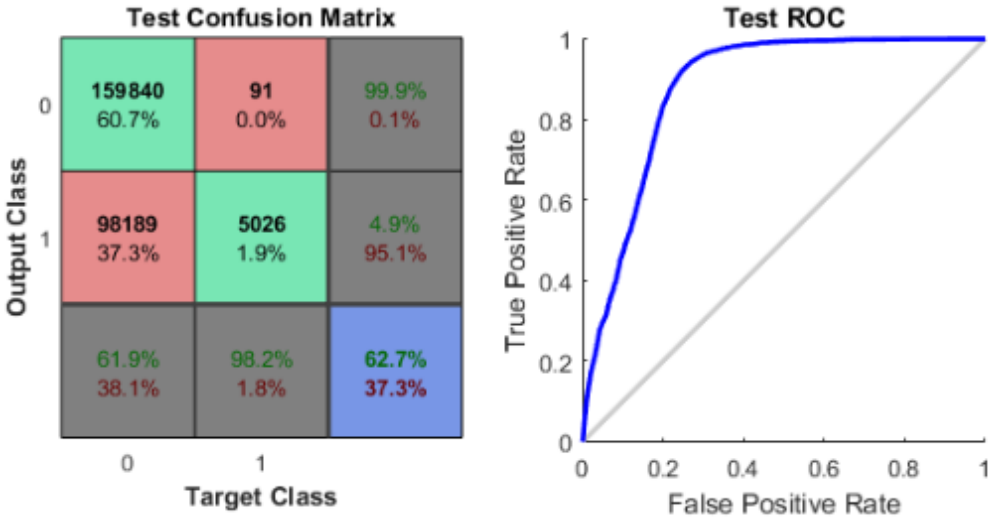
C. Entrenamiento y validación: 'GT_CenterBias_2575'. Test: media.



D. Entrenamiento y validación: 'GT_CenterBias_5050'. Test: media.



E. Entrenamiento y validación: 'GT_CenterBias_2575'. Test: moda.

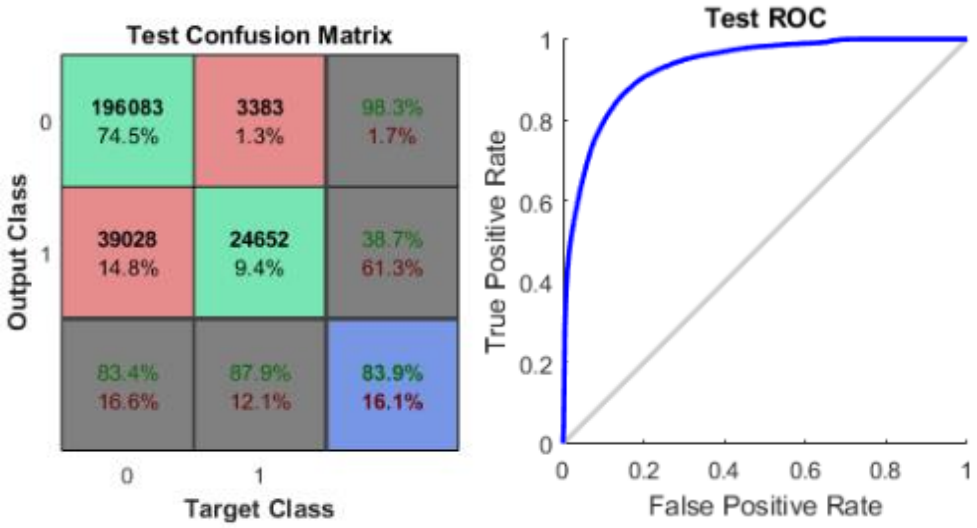


F. Entrenamiento y validación: 'GT_CenterBias_5050'. Test: moda.

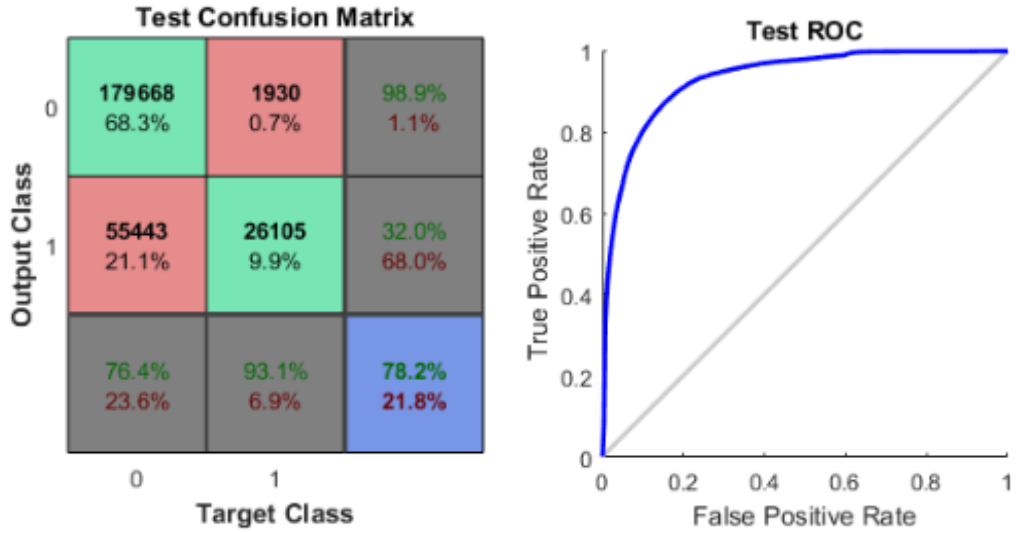
Figura 22. *Resultados del entrenamiento según la base de datos, para distintos conjuntos de test. En todas las gráficas se muestran los resultados usando todos los descriptores*

Asimismo, el diezmado de las salidas negativas, que hace que las bases de datos tengan una proporción equilibrada de positivos y negativos, elimina la tendencia que tenía la red neuronal en el anterior experimento, de clasificar todas las entradas como segmentos no salientes. Esto puede observarse con más claridad en las matrices de confusión de la Figura 22 desde (C) a (F). Sin embargo, al usar unos datos de evaluación obtenidos mediante la media y la moda, la mayor parte de los segmentos no son salientes, con lo que se dan numerosos falsos positivos.

Los mejores resultados, por tanto se pueden ver en la Figura 22 (A) y (B), no obstante en (A) el porcentaje de falsos negativos y positivos es menor (9%) que en (B) (14,9%). Estos buenos resultados se deben a la eliminación de los *outliers* en el conjunto de entrenamiento, que reduce la cantidad de errores obtenidos para la red neuronal de la figura 21 (A), en la que también se aplica la suma de las matrices de salida, pero con presencia de *outliers*.



A. Entrenamiento y validación: 'GT_Caras_2575'. Test: suma.

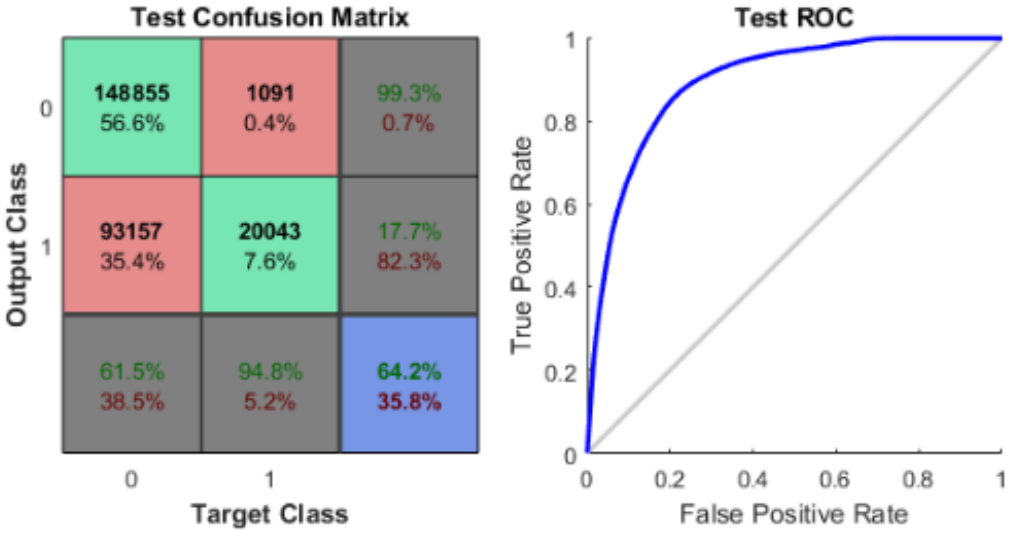


B. Entrenamiento y validación: 'GT_Caras_5050'. Test: suma.

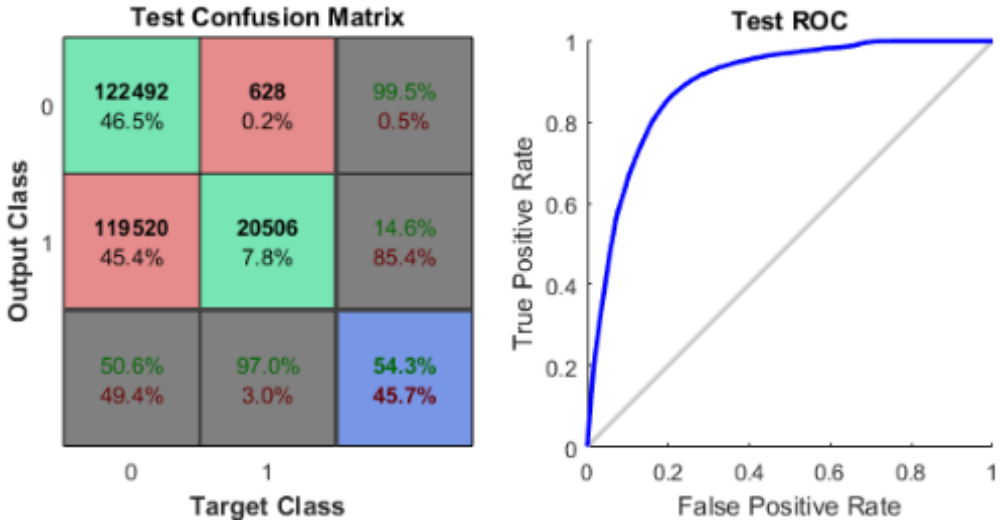
Figura 23. *Resultados del entrenamiento según la base de datos, para el conjunto de test obtenido mendiante la suma. En todas las gráficas se muestran los resultados usando todos los descriptores excepto el de center-bias.*

Tanto en la Figura 23 como en la *Figura 24*, se puede ver que los resultados son mejores esta vez, aunque al usar solamente las características *bottom-up*, como son la velocidad o la aceleración, se obtiene un mayor porcentaje de errores. Igual que se vio anteriormente, las bases de datos que fueron equilibradas al 25%-75%, dan lugar a mejores tasas de acierto.

De esta manera, las bases de datos elegidas para entrenar la red neuronal, con las que se evaluará el modelo de saliencia visual propuesto, son 'GT_CenterBias_2575', 'GT_Caras_2575' y 'GT_Ruiz_2575', usando el conjunto de test obtenido mediante la suma de las salidas de cada vídeo de todos los observadores. Así, en la sección 4.4 se elegirá cuál de ellas predice mejor la saliencia visual.



A. Entrenamiento y validación: 'GT_Ruiz_2575'. Test: suma.



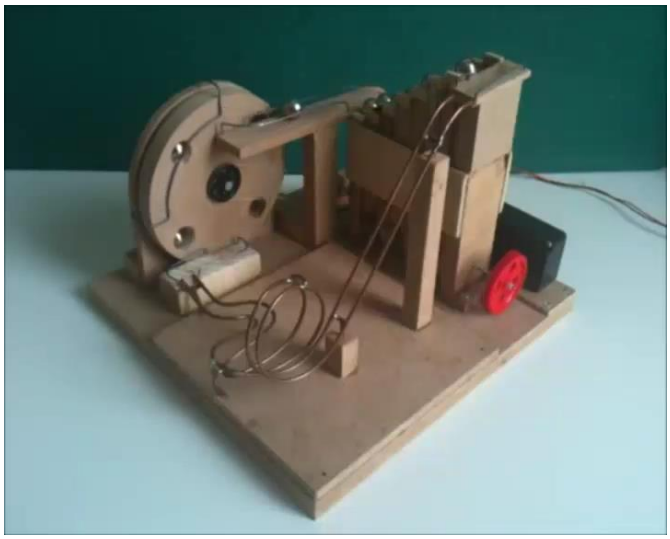
B. Entrenamiento y validación: 'GT_Ruiz_5050'. Test: suma.

Figura 24. *Resultados del entrenamiento según la base de datos, para el conjunto de test obtenido mendiante la suma. En todas las gráficas se muestran los resultados usando los descriptores extraídos por Carlos Ruiz (saliencia estática, velocidad y aceleración).*

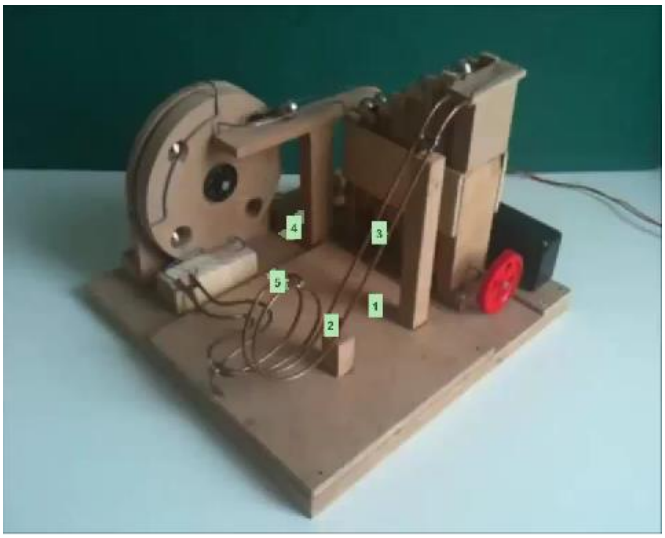
4.4 Resultados

Una vez elegidas las bases de datos, se entrena la red neuronal, obteniendo una para cada conjunto de entrenamiento –'GT_CenterBias_2575', 'GT_Caras_2575' y 'GT_Ruiz_2575'–. Estas redes neuronales han sido empleadas para predecir las fijaciones oculares en los clips 28 y 50 y, a continuación se muestran los resultados para distintos *frames* de cada vídeo.

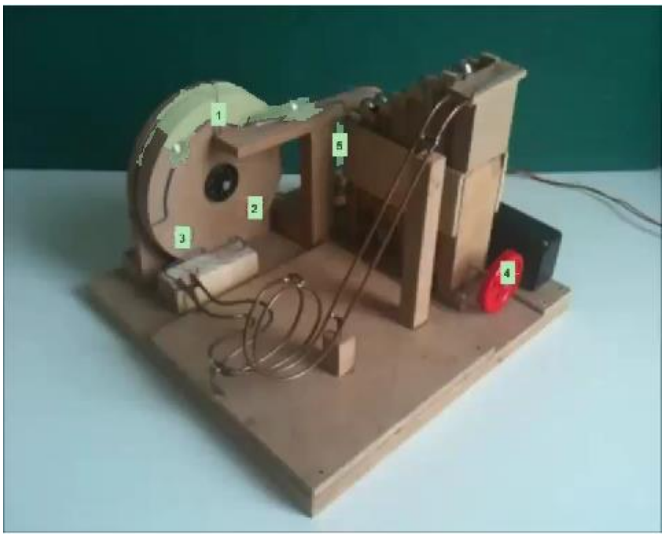
Cinco regiones salientes son etiquetadas con números del 1 al 5 en cada *frame*, de mayor a menor saliencia.



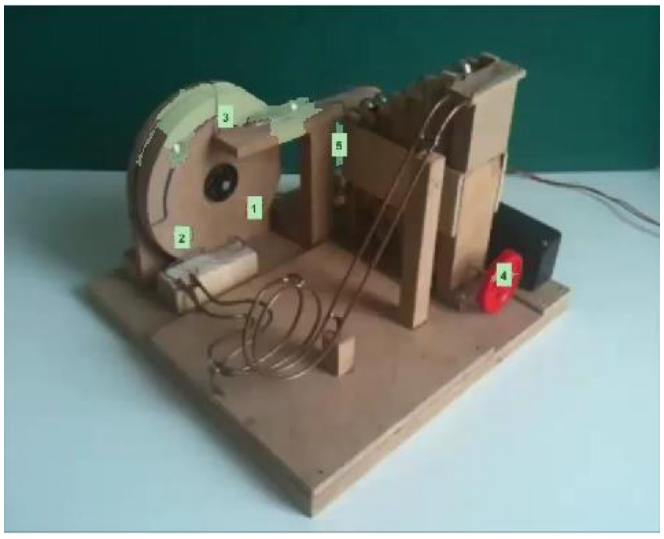
A. Clip 28, *frame* 20 original.



B. Resultados para clip 28, *frame* 20, 'GT_CenterBias_2575'.

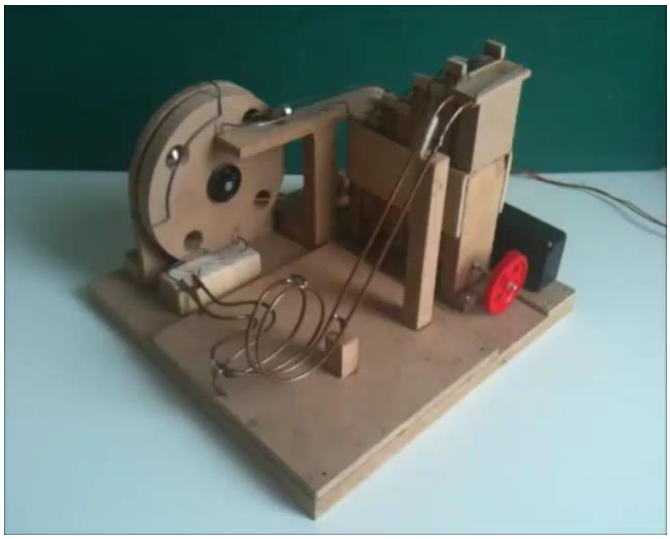


C. Resultados para clip 28, frame 20, 'GT_Caras_2575'.

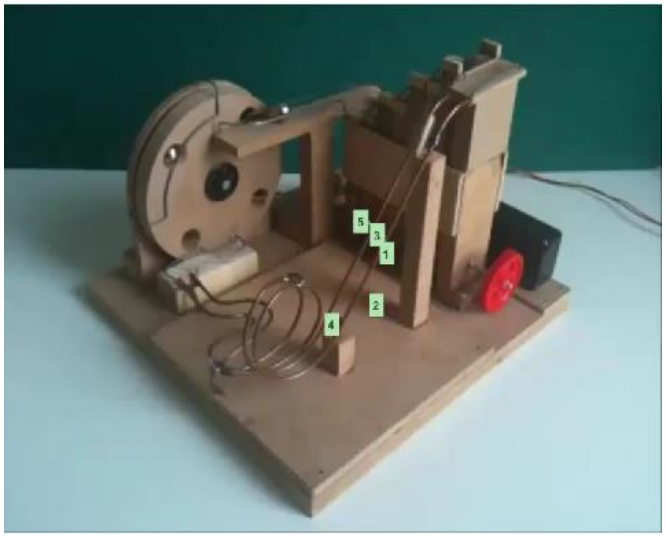


D. Resultados para clip 28, frame 20, 'GT_Ruiz_2575'.

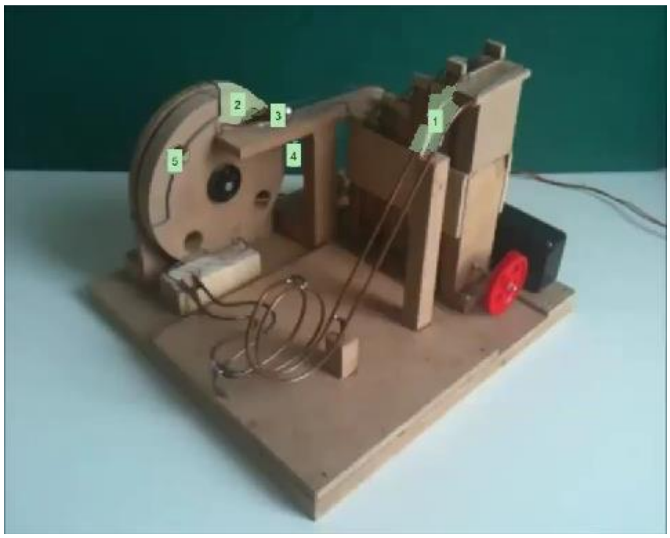
Figura 25. *Resultados para el clip 28, frame 20, usando las distintas redes neuronales.*



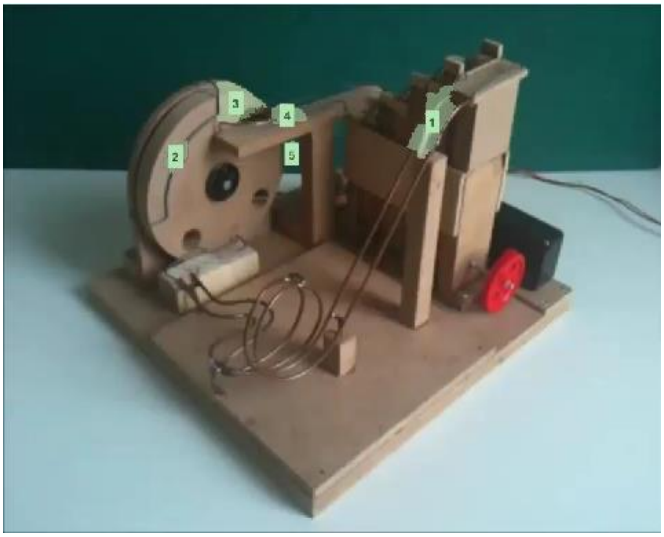
A. Clip 28, *frame* 47 original.



B. Resultados para clip 28, *frame* 47, 'GT_CenterBias_2575'.



C. Resultados para clip 28, *frame* 47, 'GT_Caras_2575'.



D. Resultados para clip 28, *frame* 47, 'GT_Ruiz_2575'.

Figura 26. *Resultados para el clip 28, frame 47, usando las distintas redes neuronales.*

En las dos figuras anteriores (Figura 25 y Figura 26), se muestra la saliencia de un vídeo que no contiene caras, en el que las ruedas giran y caen canicas por la rampa. En el *frame* 47, aunque se ve borroso debido al movimiento, está cayendo una bola por la rampa, algo que es detectado como saliente en la Figura 26 (C) y (D), pero no en (A). Esto se debe a que al añadir el descriptor de *center-bias* –unido a la seguramente baja magnitud del descriptor de velocidad–, se le otorga mucha importancia al centro de la imagen, y esto es algo que no en todos los vídeos se puede aplicar.

De este modo, en los vídeos en los que la acción principal no suceda en el centro de la imagen –que aunque son escasos pueden existir–, se obtendrán datos de saliencia erróneos. Se puede concluir que la red neuronal no ha sido capaz de obtener los pesos correctos de cada descriptor, por lo que una posible solución sería ampliar la base de datos de entrenamiento.

A. Clip 50, *frame* 5 original.

B. Resultados para clip 50, *frame* 5, 'GT_CenterBias_2575'.

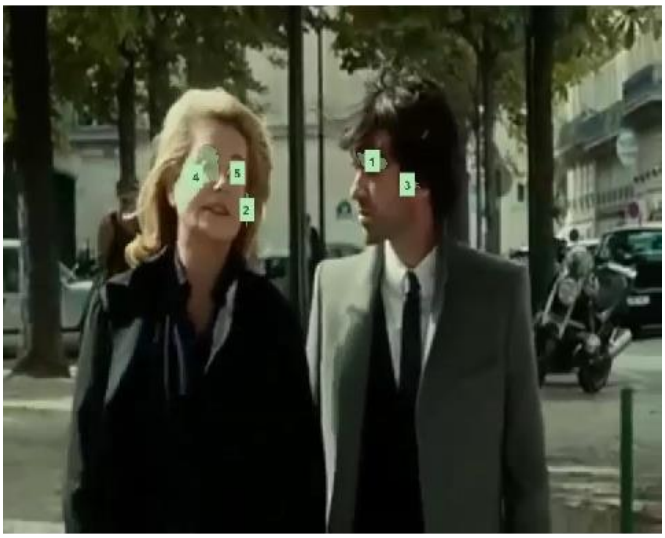
C. Resultados para clip 50, *frame* 5, 'GT_Caras_2575'.

A. Resultados para clip 50, *frame* 5, 'GT_ Ruiz _2575'.

Figura 27. *Resultados para el clip 50, frame 5, usando las distintas redes neuronales.*

A. Clip 50, *frame* 28 original.

B. Resultados para clip 50, *frame* 28, 'GT_CenterBias_2575'.



C. Resultados para clip 50, *frame* 28, 'GT_Caras_2575'.

D. Resultados para clip 50, *frame* 28, 'GT_ Ruiz _2575'.

Figura 28. *Resultados para el clip 50, frame 28, usando las distintas redes neuronales.*

En la Figura 27 se puede ver que un hombre pasa por detrás de los protagonistas de la escena. Esto hace que el foco de atención pase a este movimiento, correctamente detectado por las tres aproximaciones.

Sin embargo, si se analiza un vídeo que contiene caras, los problemas surgen cuando se emplea 'GT_Ruiz_2575', que no contiene el descriptor facial. En la Figura 28 (D) se puede ver este efecto, la que ninguno de los rostros es detectado como saliente. Así, quedan demostrados los beneficios de emplear un detector facial, ya que 'GT_Caras_2575' permite la detección de saliencia en vídeos que contienen y no contienen rostros.

Por tanto, se concluye que el conjunto de entrenamiento más eficaz es el que tiene las siguientes características –llamado 'GT_Caras_2575':

a) Descriptores de saliencia estática, velocidad, aceleración y existencia de caras.

b) *Outliers* eliminados con un umbral del 60% y radio de visión igual a 100 píxeles.

c) Muestras negativas diezmadas al 75%, obteniendo un porcentaje de positivos igual al 25%

d) 20 neuronas en la capa oculta del MLP.

Los vídeos con los resultados están disponibles en la siguiente carpeta de Drive: https://drive.google.com/folderview?id=0B46yauCXzcEzZzJ3LW9zbTNQS0k&usp=sharing

Chapter 5

Conclusions

5.1 Conclusions

There is a large number of bottom-up saliency models, but in this project we wanted to go one step further and develop a hybrid algorithm that model somehow the human visual attention. This has been done by introducing two feature maps to a saliency model: the existence of a face and the center-bias.

Nevertheless, we have obtained that the introduction of the center-bias leads to worst results than if we only incorporate face detection. This is because if we train the neural network with that descriptor, it is not able to obtain the correct weights for each feature map. However, we are sure that using a large training dataset would improve those results.

Despite this, we can say that we have achieved our objective of improving the dynamic saliency algorithm developed by Carlos, which did not perform well in presence of faces.

5.2 Future work

In the state of the art, numerous alternatives to carry out a visual saliency model have been explained, and there exist many more, but there is always room for further improvements. In this section, some ways to upgrade the performance of the system developed in this TFG are proposed.

Firstly, the implemented face detector is sensitive to light conditions, so it could be substituted by another more complex algorithm. Moreover, the extracted face descriptor is a binary matrix, in which values equal to 1 indicate the pixels where there is a face and 0 otherwise. It could be raplaced by a matrix that indicate the most important parts of the face, such as the mouth or the eyes, using larger values in these regions.

Secondly, the calculation of optical flow could also be refined to include camera motion compensation. In this way, the developed saliency model cannot predict correctly the eye fixations in a video that has this kind of motion.

Training the neural network has been one of the major problems encountered, due to the existence of eye tracking data for various observers in each video. Thereby, some tests to find the best way to train the network have not been performed due to the lack of time. For example, it could have been tested if the neural network has better performance using more videos.

As it was mentioned before, the eye tracking database that has been used includes different auditory conditions, so the proposed saliency model could be modified to take into account the effects of sound in human visual attention.

Finally, it has to be mentioned that the runtime is very large, so in the future the software should be optimized.

Capítulo 6

Presupuesto

En este capítulo se analiza tanto el tiempo como los costes de realización de este Trabajo Fin de Grado. Para ello, se ha propuesto una división en tareas del mismo, especificando la duración de cada una. Además, la secuenciación de estas actividades se muestra en un diagrama de Gantt.

Para el cálculo de los costes totales, se han tenido en cuenta gastos personales y materiales, plasmados en diferentes tablas. El presupuesto total de este TFG asciende a la cantidad de 14.744,15€.

6.1 Gestión del tiempo

En esta sección se van a definir las actividades llevadas a cabo para la realización del trabajo, así como su duración y secuenciación. De esta manera se obtendrá el tiempo de ejecución del proyecto necesario a la hora de realizar el cálculo de los costes, llevado a cabo en la sección 6.2.

En la siguiente tabla, se pueden ver las actividades indicadas en la sección 1.3, donde se explicó qué se llevaba a cabo en cada una, así como los elementos empleados para su realización.

Fase	Actividad	Horas	Días
1. Planteamiento inicial	1.1. Documentación	20	5
	1.2. Elección de modelos	24	6
2. Desarrollo del modelo	2.1. Modificación del software de saliencia dinámica	32	15
	2.2. Programación de nuevos descriptores	55	14
	2.3. Entrenamiento de la red neuronal	110	27
3. Evaluación del modelo	3.1. Obtención de resultados	25	5
	3.2. Análisis de resultados	10	2
4. Redacción de la memoria	4.1. Documentación	55	15
	4.2. Escritura de la memoria	90	20
TOTAL		421	109

Tabla 6. *Fases necesarias para la realización del proyecto, divididas en actividades e indicando su duración en horas y días.*

En paralelo a estas actividades, se ha realizado el seguimiento del trabajo y la dirección por parte de la tutora.

A continuación se va a emplear el *software* Project Libre para realizar el diagrama de Gantt que representa la secuencia de las actividades de la Tabla 6. Aunque estas actividades no se han llevado a cabo de manera continua, en el diagrama se va a ilustrar el tiempo total requerido si éstas se realizaran de forma ininterrumpida.

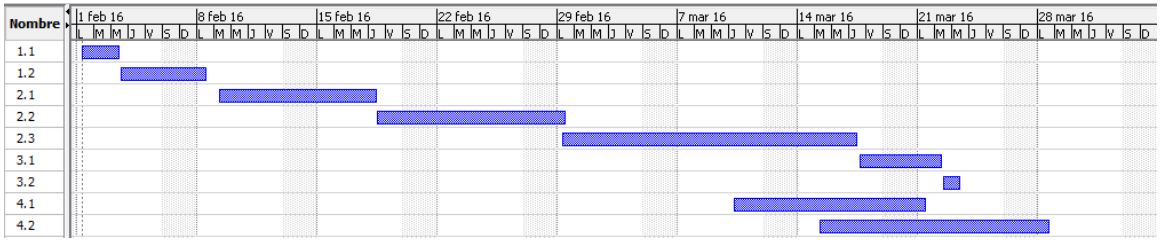


Figura 29. *Diagrama de Gantt.*

6.2 Análisis de costes

Una vez realizada la gestión del tiempo, se puede proceder a calcular el coste total de este proyecto. Por un lado, se van a tener en cuenta los costes materiales y por otro los costes de personal considerando dos tipos de perfiles: estudiante y profesor titulado.

Para el cálculo del coste imputable al equipo empleado se hace uso de la ecuación 6.1 y los datos incluidos en la Tabla 7.

EQUIPOS					
Descripción	**Coste (€)**	**% Uso dedicado al proyecto**	**Dedicación (meses)**	**Periodo de depreciación (meses)**	**Coste imputable (€)**
Ordenador portátil	600	100	2	60	20
TOTAL			20		

Tabla 7*. Costes del equipo empleado.*

$$Coste\ imputable = \frac{Dedicación}{Periodo\ de\ depreciación} \cdot Coste\ equipo \cdot \%Uso \qquad (6.1)$$

OTRO MATERIAL	
Descripción	**Coste (€)**
Licencia de uso académico de Matlab	500
Computer Vision System Toolbox	200
Neural Network Toolbox	200
TOTAL	900

Tabla 8. *Costes del material empleado.*

Según la Federación estatal sectorial de la Unión General de Trabajadores (FETE-UGT), la retribución salarial por hora de trabajo de un profesor titular son 32,31€, mientras que un estudiante cobra 18,70 € la hora[10].

PERSONAL			
Categoría	**Coste de hora (€)**	**Dedicación (horas)**	**Coste (€)**
Estudiante	18,7	421	7.872,7
Profesor titular	32,31	105	3.392,55
TOTAL			11.265,25

Tabla 9*. Retribución salarial del personal.*

[10] Tabla salarial para el año 2015: http://www.feteugt.es/Data/UPLOAD/PRI-tablas-salariales-XIII-convenio-centros-educacion-universitaria-2015.pdf

COSTE TOTAL DEL PROYECTO	
Coste de equipos	20€
Coste de otros materiales	900€
Coste de personal	11.265,25€
IVA (21%)	2.558,9€
TOTAL	14.744,15€

El presupuesto total de este proyecto asciende a la cantidad de 14.744,15€.

Leganés a 22 de junio de 2016.

La ingeniera proyectista.

Fdo. Laura López Aragoneses

Appendix A

Abstract

1. Introduction

Nowadays, most saliency models are focused on the extraction of bottom-up information, like color, contrast or motion, to predict the eye fixations of an observer. Nevertheless, human visual attention focuses on high-level features of the image, which provide relevant information in order to understand the scene –top-down attention–.

A small amount of visual saliency algorithms include this top-down attention, so our purpose is to improve the performance of an existing bottom-up saliency model, by adding some high-level features: the tendency of looking to the center of the screen and the fact that we pay more attention to faces.

In order to build this hybrid model, we are going to start from a dynamic bottom-up saliency algorithm, developed by Carlos Ruiz in his Final Year Project[46] (in Spanish, *Trabajo Fin de Grado*, TFG). So, the main objective is to improve its performance.

On the other hand, enlarging the eye tracking database was proposed as future work in his TFG, so other objective is to accomplish this task. Although a large database formed by 60 videos with eye fixations data was found, only 8 of them have been used due to the time restrictions.

To achieve the objectives, some steps have been followed. They are briefly explained below, together with the elements used in each one of them.

1. **Initial approach:** the purpose of this first step was to understand the task to be tackled and delve into it. In addition, the first decisions on the algorithms that have been used were taken.

2. **Algorithm development:** this phase can be divided into three steps.

    a. Dynamic saliency software adjustment: the saliency software[46] that has been improved, reads the eye fixations data from a *.xml* file. However, the eye tracking database[11] that have been used was built in Matlab, so the software had to be adapted.

    b. Incorporation of top-down cues: two important characteristics of human visual attention have been included in the saliency algorithm, as it was mentioned above: the tendency of looking to the center of the screen and the fact that we pay more attention to faces. Both have been developed in Matlab, the first by introducing a bias proportional to a Gaussian distribution, situated in the center of the images (section 3.2.2) and the second using the Viola-Jones face detection algorithm, implemented in Matlab's *Computer Vision System Toolbox*[12] (section 3.2.1).

    c. Neural network training: firstly, the software to accomplish this task was chosen. In this case, the Matlab's *Neural Network Toolbox*[13] has been used (section 3.2.3). Then, various training sets have been built as is explained in 4.2, in order to find the best way to train the network.

3. **Algorithm evaluation:** to evaluate the modified saliency model, two videos have been tested with different training sets.

4. **Report writing:** this last step has been carried out by looking up several research papers, listed in the references section. Microsoft Word has been used as an editing tool.

2. State of the art

There exist a large number of algorithms that seek to predict the eye fixations on images, in other words, its visual saliency. This field of research started with a saliency model proposed by Itti et al.[22], and now there exist a large number of prediction systems, of which few of them incorporate top-down[28].

On the one hand, bottom-up saliency models are based on visual characteristics of an image, as color or contrast. On the other hand, top-down models include high-level

[11] Database 1, *Dynamic natural scenes*: http://antoinecoutrot.magix.net/public/databases.html
[12] *Computer Vision System Toolbox*: http://es.mathworks.com/products/computer-vision/
[13] *Neural Network Toolbox*: http://es.mathworks.com/products/neural-network/

features that allow object and scene identification. Although most eye movements are driven by these features, many models are bottom-up because of the simplicity of extracting low-level characteristics.

In this manner, it was decided to modify the dynamic saliency algorithm, developed by Carlos Ruiz, in order to include two top-down attention aspects: the tendency of looking to the center of the screen and to the existing faces in an image, implemented with a face detector.

This section will list different algorithms of visual saliency and face detection, focusing on the ones that have been used.

2.1 Visual saliency

Nowadays, there are static and dynamic saliency models. The first are numerous and seek to predict the fixations in a static image, while the others take into account the motion of the objects in a scene, in addition to the spatial features.

In 1985, Koch and Ullman[27] introduced the definition of static saliency map, and since then there have been many spatial saliency model variations developed by several researchers. The most popular was implemented by Itti et al.[22], which is based on bottom-up saliency. Its operation can be explained in four steps:

a) Feature extraction: firstly, input image is rescaled forming a Gaussian pyramid. Then, intensity, color and orientation are extracted from each level of the pyramid. In this way, there is a map for every feature in each level. Finally, center-surround differences are calculated.

b) Feature maps calculation: in this step, the previously computed maps are normalized and combined to give a single map for each characteristic.

c) Saliency map construction: now, the saliency map is calculated by merging all feature maps.

d) Salient region detection: to locate the region of the map where the saliency is more prominent, a computational principle called *winner-take-all* is utilized.

Liu et al.[35] proposed an improvement of this model, based on a single feature –the contrast–, including image segmentation.

Some spatial saliency algorithms that include top-down attention, were proposed by Bruce and Tsotsos[7] (Attention based on Information Maximization, AIM), Hou and Zhang[20] (Dynamic Visual Attention, DVA) and Zhang et al.[63] (Saliency Using Natural statistics, SUN).

On the other hand, there exist dynamic saliency models that extract scene features such as velocity and acceleration. The calculation of the optical flow is the most common approach to carry out these models. This concept was introduced by Horn and Schunk in 1981[19] and enhanced by Black and Anandan[4].

The dynamic saliency model that we have modified, was implemented by Carlos Ruiz[46] using the algorithm of Itti et al. for extracting the spatial saliency, and calculating the optical flow by Liu's model.

2.2 Face detection

To introduce the existence of faces in a frame as a high-level feature, a face detector based on Viola-Jones algorithm[55] is going to be used. In this section, some face detection models are listed, although the Viola-Jones is explained in depth.

2.2.1. Viola-Jones algorithm

This model was the first real-time face detector implemented and it consists of three steps:

a) Feature extraction: Viola and Jones[55] used three types of Haar-like templates –also called basis– to obtain different features, placing them in different regions of the image to obtain the values. More than 160.000 features can be obtained using a base of 24x24 pixels. For this reason, a new image representation called integral image was introduced, which allows a rapid extraction of these characteristics. In that image, each pixel has the value of the sum of all the pixels above and to the left.

b) AdaBoost learning algorithm: to process all of those features, Viola and Jones used a modified AdaBoost algorithm, in order to select the most important characteristics to detect a face. This is a machine learning boosting system, which combines a set of weak classifiers to construct a strong one.

c) Cascade of classifiers: this cascade is formed by the strong classifiers obtained above, and is in charge of reducing the runtime.

2.2.2 Other face detection algorithms

There exist a large number of face detection algorithms developed in different ways. In this section, three types of facial detectors are going to be treated, but there are many more.

On the one hand, there are face detection algorithms based on rigid templates, as the Viola-Jones[55] model. All of them are based on learning these templates using a cascade of classifiers. Sakai et al.[47], Craw et al.[3] or Samal et al.[48] proposed some of those models.

On the other hand, systems based on deformable templates were also developed by many authors such as Yuille et al.[60] –who introduced this concept–, Kwon[29] or Lanitis et al.[31]. In this case, the templates are able to change their parameters to better fit the image.

Another type of face detection algorithms includes those that are implemented using neural networks. This kind of detector was first proposed by Féraund et al.[14] in 2001.

Since then, numerous approximations have been proposed (Chen et al.[9] or Zhang et al.[62]).

3. Description of the model

The spatio-temporal saliency model developed by Carlos Ruiz[46] in Matlab, extracts three different features of each pixel: the static saliency, the velocity and the acceleration. The objective of this project is to include two more descriptors, in order to model somehow the visual attention: one indicating which regions have a face and another one that emphasizes the importance of the center of the image.

After all image features are extracted, a neural network of type MultiLayer Perceptron (MLP) is trained to classify the pixels as salient or not. To do this, the network needs inputs –image features– and outputs or objectives –if the feature leads to a salient region–. These outputs are obtained by using an eye tracker, which collects the eye fixations of an observer when watching a video.

In this section, the dynamic saliency model from which we depart and the modifications that have been applied to it will be explained, in addition to the neural network training software selected.

3.1 Dynamic saliency model

This model is composed by three elements:

a) Spatial saliency calculation: to accomplish this task, the *SaliencyToolbox* of Walther and Koch[56] was used. It is based on the Itti et al.[22] static saliency model. This toolbox has a simple graphical user interface that allows the modification of various parameters, such as the number of levels of the Gaussian pyramid or the iterations of normalization.

b) Optical flow computation: as it was explained in the state of the art, in order to compute the dynamic saliency, an algorithm to compute the optical flow is needed. In this case, the approach developed by Liu[34] was selected, due to its good results detecting small displacements.

c) Image segmentation: this element was included for two reasons. The first was the inaccuracy of both the static saliency model and the calculation of the optical flow. In this way, all the pixels belonging to a segment have the same feature maps. Secondly, it was also included to compensate the misbehavior of the eye tracker, used to obtain the database that was utilized by Carlos Ruiz to train the neural network.

3.2 Modifications to the dynamic saliency model

The first change in the dynamic saliency model carried out was to adapt it to the new eye tracking database that has been used[12]. Carlos Ruiz made use of data stored in an *.xml* file, but now the eye fixations are in a *.mat*.

Then, the facial descriptor was incorporated by using a face detector. It has to be implemented in Matlab, so we have decided to use the *Computer Vision System Toolbox* of Matlab. It includes the system called *vision.CascadeObjectDetector*, based on Viola-Jones algorithm, which is simple and allows the front and side face detection. It also allows the detection of important parts of the face such as the mouth or the eyes, but after some preliminary tests, we have decided that the best way to determine the location of a face is by using the combination of profile and front detection.

This descriptor is obtained by analysing each frame in order to find faces. Then, a binary matrix is calculated, in which the pixels that do not contain a face are labeled with 0 and otherwise with 1. An example can be seen in the next figure:

A. Original frame. **B.** Face descriptor.

Figure A1. *Calculation of the facial descriptor.*

Finally, three center-bias descriptors have been added, introducing a bias proportional to a Gaussian distribution, situated in the center of the images. Each one has a different standard deviation as can be seen in the figure. In this way, more importance is given to the center of the screen.

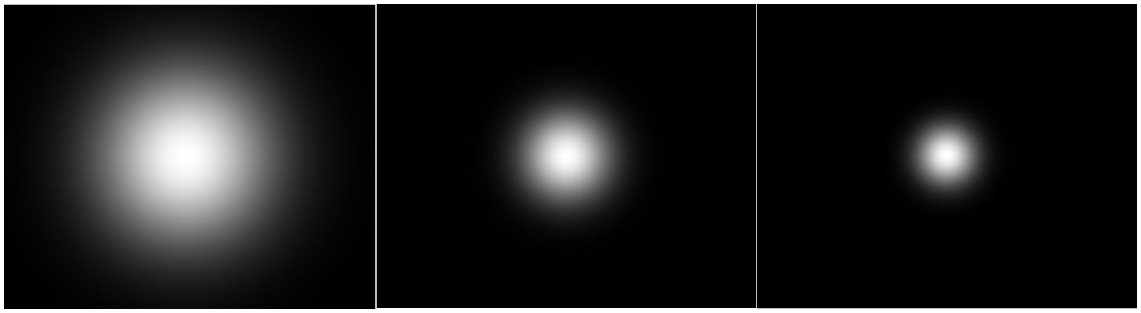


Figure A2. *Center-bias descriptors for different standard deviation values.*

To train the neural network the Neural Network Toolbox, implemented in Matlab, has been use, because of its simplicity. It was configured to resolve a classification problem –if a region is salient or not–, so a MultiLayer Perceptron was used.

4. Experiments

To carry out the experiments, a database formed by videos that have eye tracking information has been used. This database was collected by Coutrot and Guyader[12] and it has 60 videos of which we are going to use 8 –6 to train and validate the network and 2 to test it–, due to the lack of time. Some of these videos contain faces and others do not, but all of them have eye fixation data of various observers.

In order to train the neural network, three sets have to be constructed using that database: a training, validation and test set. To do this, the 8 videos have to be processed in this way:

a) Feature maps extraction: the features –static saliency, velocity, acceleration, faces and center-bias– have to be calculated for every pixel of each frame.

b) Frame segmentation: all the frames are segmented. Then, the mean of each characteristic of the pixels of a segment is calculated. In this manner, all the pixels of a segment have the same descriptors. These values are the inputs of the neural network and are equal for all the observers of a video.

c) Gaze radius determination: eye tracking data is read from the database, and a circumference of radius R and center located at the coordinates of the fixation, is stablished to decide which segments are salient. If a segment has some pixel inside this circle, it is said to be salient. These segments are represented by a 1, while the rest has a value of 0, forming an output matrix for each observer –depending on the eye fixations of each one–.

Once those steps have been carried out, two different ways of calculating the training set were proposed.

The first was to build a common matrix of inputs and outputs for all the observers of each video. The inputs are the same for each one, but the outputs are combined by calculating the sum, mean or mode. This approach was not succesfull, due to the high percentage of false positives and negatives, so another one was proposed.

The second way was to remove the outliers, that is, all the segments that are not clearly salient or not salient. So a threshold was applied to the descriptors, in order to reduce the number of false positives and negatives. In this case, we concatenate the inputs and outputs of all observers and videos in a single matrix, instead of calculating a common one. As the number of negative outputs was very large, we balanced the matrix in order to obtaining two diferent sets: one with 75% of negative outputs and another one with 50%.

Using this approach, we build the training sets of the table:

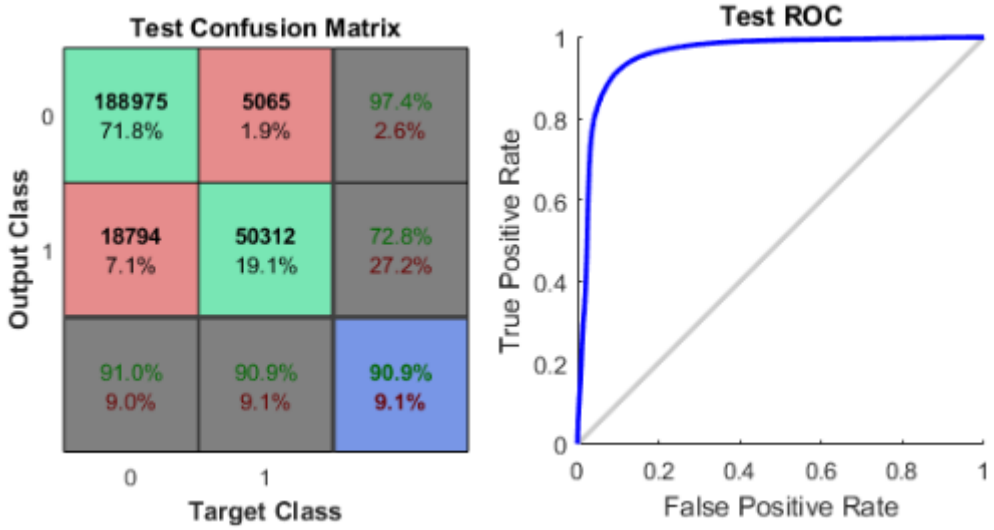
Name	Feature maps	Positives	Negatives	Total segments
GT_Ruiz_2575	Static saliency, velocity and acceleration	25%	75%	604.016

GT_Ruiz_5050		50%	50%	302.008
GT_Caras_2575	Static saliency, velocity, acceleration and faces	25%	75%	1.078.924
GT_Caras_5050		50%	50%	539.462
GT_CenterBias_2575	Static saliency, velocity, acceleration, faces and center-bias	25%	75%	1.810.888
GT_CenterBias_5050		50%	50%	905.444

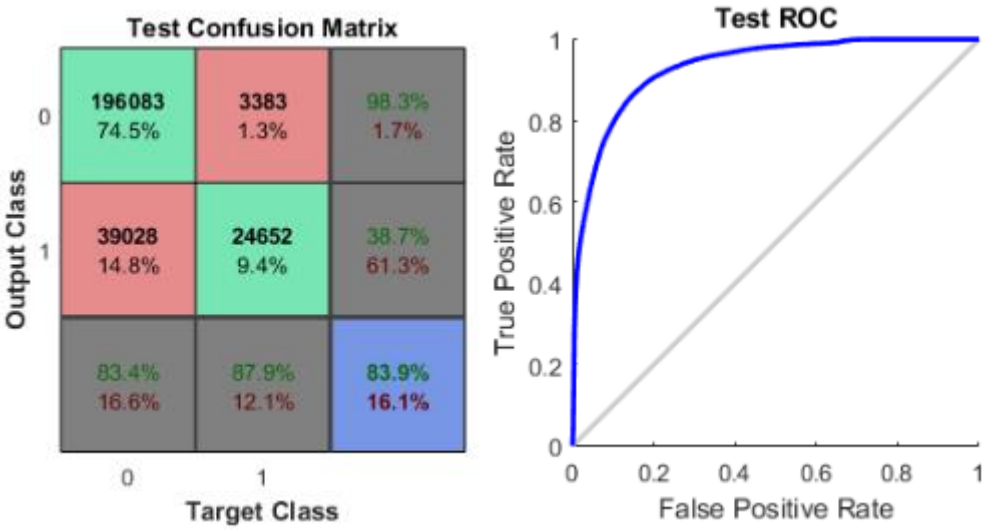
Table A1. *Different training sets and their characteristics.*

Those databases were built using 6 videos in order to train and validate the neural network. Now, the test set has to be constructed with the other 2 videos. To do this, we have removed the outliers, but instead of concatenating the matrices, we have built a common one for all the observers of each video, using the sum, the mean and the mode. This has been done beacause it is not correct to test the network with different outputs for a same segment.

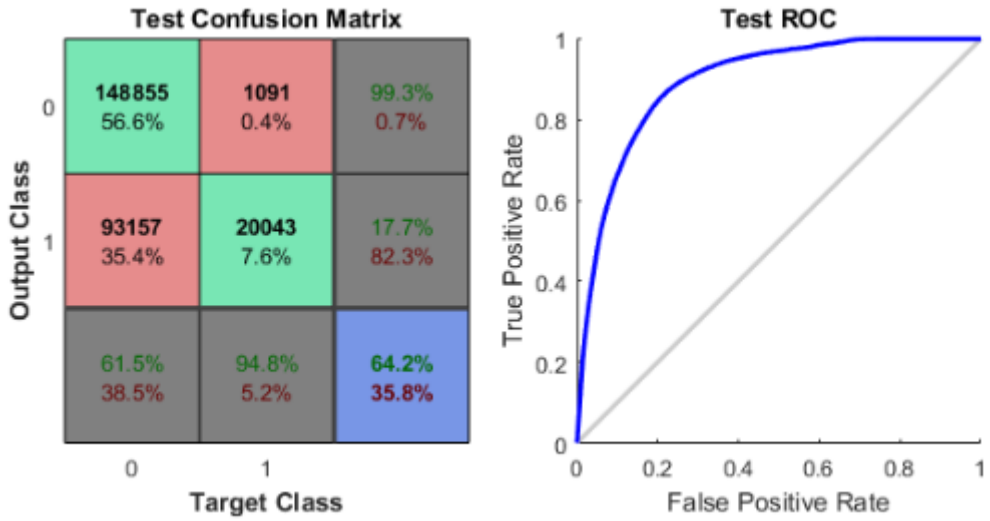
When training the MLP with those databases and 20 neurons in its hidden layer, we have obtained that the best way of doing this is using 'GT_Ruiz_2575', 'GT_Caras_2575' and 'GT_CenterBias_2575', and the testing set calculated by the sum.



A. Training and validation: 'GT_CenterBias_2575'. Test: sum.



B. Training and validation: 'GT_Caras_2575'. Test: sum.



C. Training and validation: 'GT_Ruiz_2575'. Test: sum.

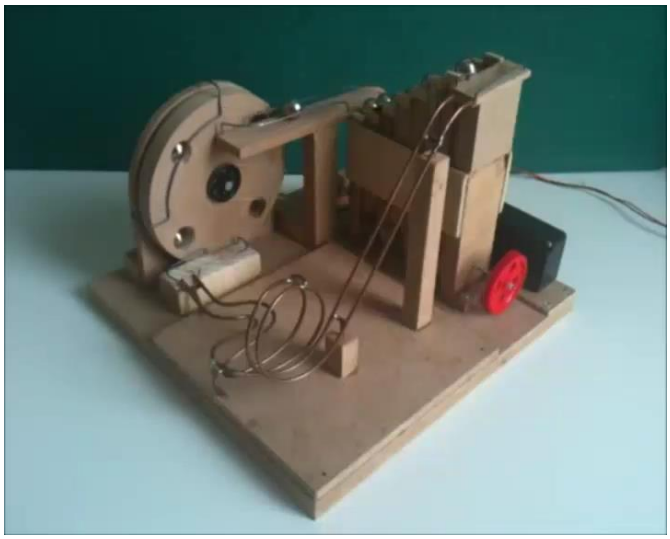
Figure A3. *Training results depending on the ground truth.*

4.1 Results

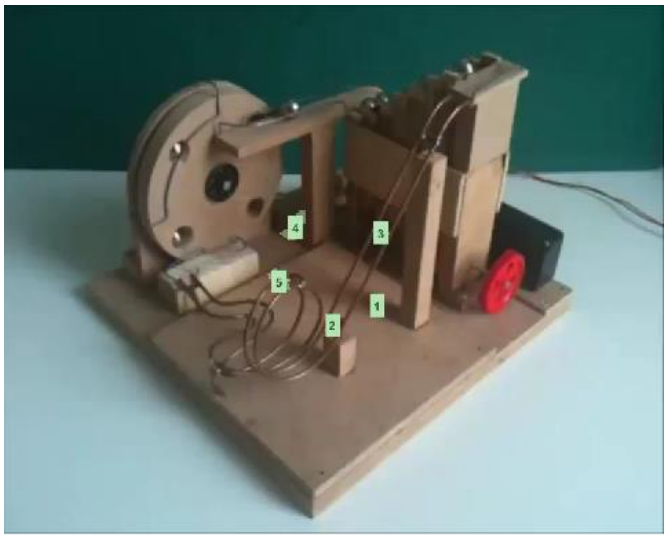
Once the ground truth datasets have been selected, we can train the neural network in order to evaluate clips 28 and 50 of the eye tracking database.

We have obtained that using the center-bias descriptor the neural network does not perform as it was expected. These results can be seen in Figure A4 (B), where only the center of the image has been detected as salient, when the wheels and the balls are in motion.

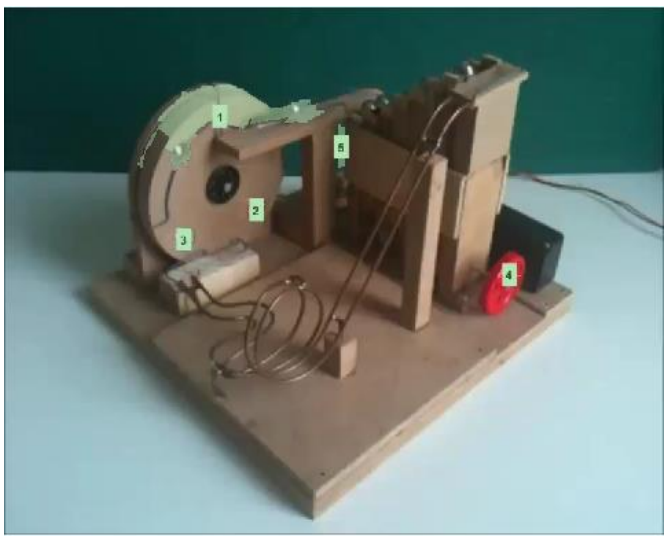
In counterpart, introducing the face detector has improved the results of the dynamic saliency algorithm developed by Carlos (Figure A5, (C) and (D)).



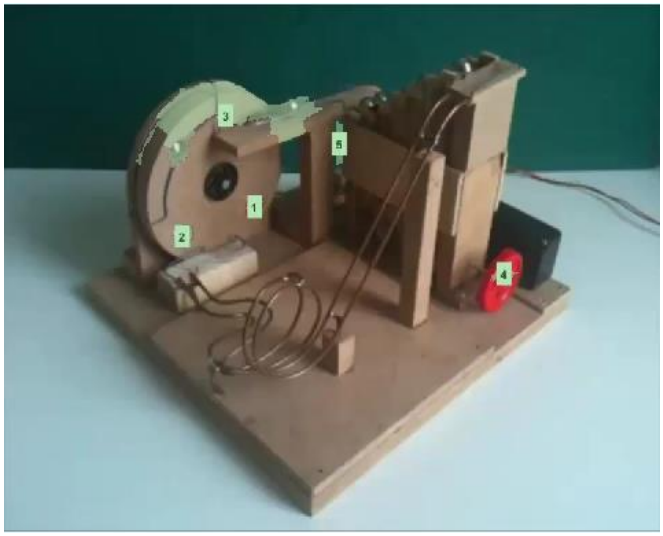
A. Clip 28, *frame* 20.



B. Results for clip 28, *frame* 20, ‘GT_CenterBias_2575’.



C. Results for clip 28, frame 20, ‘GT_Caras_2575’.

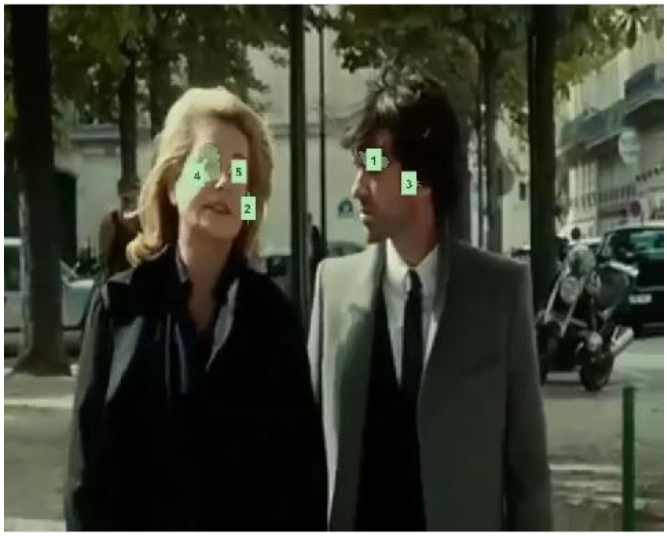


D. Results for clip 28, frame 20, 'GT_Ruiz_2575'.

Figure A4. *Results for clip 28, frame 20.*

A. Clip 50, *frame* 28.

B. Results for clip 50, *frame* 28, 'GT_CenterBias_2575'.



C. Results for clip 50, *frame* 28, 'GT_Caras_2575'.

D. Results for clip 50, *frame* 28, 'GT_ Ruiz _2575'.

Figure A5. *Results for clip 50, frame 50.*

These results can be seen at:

https://drive.google.com/folderview?id=0B46yauCXzcEzZzJ3LW9zbTNQS0k&usp=sharing

5. Conclusions

There is a large number of bottom-up saliency models, but in this project we wanted to go one step further and develop a hybrid algorithm that model somehow the human visual attention. This has been done by introducing two feature maps to a saliency model: the existence of a face and the center-bias.

Nevertheless, we have obtained that the introduction of the center-bias leads to worst results than if we only incorporate face detection. This is because if we train the neural network with that descriptor, it is not able to obtain the correct weights for each feature map. However, we are sure that using a large training dataset would improve those results.

Despite this, we can say that we have achieved our objective of improving the dynamic saliency algorithm developed by Carlos, which did not perform well in presence of faces.

Referencias

[1] Achanta, R., Hemami, S., Estrada, F. y Susstrunk, S., *Frequency-tuned Salient Region Detection*. Computer Vision and Pattern Recognition, 1597-1604, 2009. Doi: 10.1109/CVPR.2009.5206596. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5206596&isnumber=5206488

[2] Bindemann, M., *Scene and Screen Center Bias Early Eye Movements in Scene Viewing*. Vision Research, 50(23): 2577-2587, 2010, ISSN 0042-6989. Doi: 10.1016/j.visres.2010.08.016. URL: http://www.sciencedirect.com/science/article/pii/S0042698910004025

[3] Biskupska, M., *Bottom-Up Saliency Maps – A Review*. 2013. URL: http://www.imm.org.pl/imm/plik/elektronika2013-07-2013_nn256.pdf

[4] Black, M. J. y Anandan, P., *The Robust Estimation of Multiple Motions: Parametric and Piecewise-Smooth Flow Fields*. Computer Vision and Image Understanding, 63(1): 75-104, 1996. Doi: doi:10.1006/cviu.1996.0006. URL: http://www.sciencedirect.com/science/article/pii/S1077314296900065

[5] Borji, A. e Itti, L., *State-of-the-Art in Visual Attention Modeling*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(1), 2013. Doi: 10.1109/TPAMI.2012.89 URL: http://ilab.usc.edu/publications/doc/Borji_Itti13pami.pdf

[6] Brox, T. Bruhn, A., Papenberg N. y Weickert J., *High Accuracy Optical Flow Estimation Based on a Theory for Warping*. European Conference on Computer Vision, 2024: 25-36, 2004. URL: http://lmb.informatik.uni-freiburg.de/Publications/2004/Bro04a/

[7] Bruce, N. D. B. Y Tsotsos, J. K., Saliency, *Attention and Visual Search: An Information Theoretic Approach*. Journal of Vision, 9(5), 2009. Doi: 10.1167/9.3.5. URL: http://jov.arvojournals.org/article.aspx?articleid=2193531

[8] Bruhn, A., Weickert, J. y Schnörr, C., *Lucas/Kanade Meets Horn/Schunck: Combining Local and Global Flow Methods*. IEEE International Journal of Computer Vision, 61(3): 211-231, 2005. Doi: 10.1023/B:VISI.0000045324.43199.43. URL: https://www.researchgate.net/publication/220660118_LucasKanade_Meets_Horn Schunck_Combining_Local_and_Global_Optic_Flow_Methods

[9] Chen Y.-N., Han, C.-C., Jeng, B.-S., Fan, K.-C., *A CNN-Based Face Detector with a Simple Feature Map and a Coarse-to-Fine Classifier*. IEEE Transactions on Pattern Analysis and Machine Intelligence, PP(99). 2009. Doi: 10.1109/TPAMI.2007.70798. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4378386&isnumber=4359286

[10] Cheng, M.-M., Warrel, J., Lin, W.-Y., Zheng, S., Vineet, V. y Crook, N., *Efficient Salient Region Detection with Soft Image Abstraction*. IEEE International Journal of Computer Vision, 1529-1536, 2013. Doi: 10.1109/ICCV.2013.193. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6751300

[11] Cootes, T. F. y Taylor, C. J., *Locating Faces Using Statistical Feature Detectors*. Automatic Face and Gesture Recognition, 204-209, 1996. Doi: 10.1109/AFGR.1996.557265. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=557265&isnumber=12122

[12] Coutrot, A. y Guyader, N., *Toward the Introduction of Auditory Information in Dynamic Visual Attention Models*. 2013. URL: http://antoinecoutrot.magix.net/public/assets/coutrot_et_al_wiamis2013.pdf

[13] Craw, I., Ellis, H. y Lishman, J.R., *Automatic extraction of face-features*. Pattern Recognition Letters, 5(2): 183-187, 1987, ISSN 0167-8655. Doi: 10.1016/0167-8655(87)90039-0. URL: http://www.sciencedirect.com/science/article/pii/0167865587900390

[14] Féraund, R., Bernier, O. J. Viallet J-E. y Collobert, M., *A Fast and Accurate Face Detector Based on Neural Networks*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(1), 42-53, 2001. Doi: 10.1109/34.899945. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=899945&isnumber=19479

[15] Girshick, R., Donahue, J., Darrel, T. y Malik, J., *Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation*. 2013. URL: http://arxiv.org/abs/1311.2524

[16] Govindaraju, V., Sher, D. B., Srihari, R. K. y Srihari, S. N., *Locating human faces in newspaper photographs*. Computer Vision and Pattern Recognition, 549-554, 1989. Doi: 10.1109/CVPR.1989.37900. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=37900&isnumber=1531

[17] Harel, J., Koch, C. y Perona, P., *Graph-Based Visual Saliency*. Proceedings of Neural Information Processing Systems, 2006. URL: https://papers.nips.cc/paper/3095-graph-based-visual-saliency.pdf

[18] Henderson, J. M. y Hollingworth, A., *High-Level Scene Perception*. Annual Reviews of Pyschology, 50: 243-71, 1999. URL: http://jhenderson.org/vclab/PDF_Pubs/Henderson_Hollingworth_AnnRev_1999.pdf

[19] Horn, B. K. P. y Schunk, B. G, *Determining Optical Flow*. Artificial Intelligence 17(1-3): 185-203, 1981, ISSN 0004-3702. Doi: 10.1016/0004-3702(81)90024-2. URL: http://www.sciencedirect.com/science/article/pii/0004370281900242

[20] Hou, X. y Zhang, L., *Dynamic Visual Attention: Searching for coding length increments*. Advances in Neural Information Processing Systems 21, 2009. URL: http://resolver.caltech.edu/CaltechAUTHORS:20160329-154744056

[21] Hou, X., Zhang, L., *Saliency Detection: A Spectral Residual Approach*. IEEE International Journal of Computer Vision, 1-8, 2007. Doi: 10.1109/CVPR.2007.383267 . URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4270292

[22] Itti, L., Koch, C. y Niebur, E., *A Model of Saliency-Based Visual Attention for Rapid Scene Analysis*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(11): 1254-1259, 1998. Doi: 10.1109/34.730558. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=730558&isnumber=15773

[23] Jensen O. H., *Implementing the Viola-Jones Face Detection Algorithm*. 2008, ISBN 87-643-0008-0, ISSN 1601-233X. URL: http://etd.dtu.dk/thesis/223656/ep08_93.pdf

[24] Judd, T., Ehinger, K., Durand, F. y Torralba, A., *Learning to Predict Where Humans Look*. IEEE International Journal of Computer Vision, 2106-2113, 2009. Doi: 10.1109/ICCV.2009.5459462. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5459462

[25] Keren, D., Osadchy, M., Gotsman, C., *Antifaces: a Novel Fast Method for Image Detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(7): 747-761. 2001. Doi: 10.1109/34.935848. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=935848&isnumber=20256

[26] Kirby, M., Sirovich, L., *Application of the Karhunen-Loève Procedure for the Characterization of Human Faces*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(1): 103-108, 1990, ISSN: 0162-8828. Doi: 10.1109/34.41390. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=41390&isnumber=1584

[27] Koch, C., Ullman, S., *Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry*. Human neurobiology, 4(4): 219-27, 1985. Doi: 10.1007/978-94-009-3833-5_5. URL: https://www.researchgate.net/publication/19324926_Shifts_in_Selective_Visual_Attention_Towards_the_Underlying_Neural_Circuitry

[28] Kümmerer, M., Wallis, T. S. A. y Bethge, M., *Information-theoretic model comparison unifies saliency metrics.* 2015. Doi: 10.1073/pnas.1510393112. URL: http://www.pnas.org/content/112/52/16054.abstract?sid=8c13d38e-4b3f-4bcf-829e-81905b2f6f27

[29] Kwon, Y. H. y da Vitoria Lobo, N., *Face Detection Using Templates*. Pattern Recognition, 1: 764-767, 1994. Doi: 10.1109/ICPR.1994.576435. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=576435&isnumber=12513

[30] Lam, K.-M. y Yan, H., *Fast Algorithm for Locating Head Boundaries*. Journal of Electronic Imaging, 3(4), 351-359, 1994. Doi: 10.1117/12.183806. URL: http://spie.org/Publications/Journal/10.1117/12.183806

[31] Lanitis, A., Taylor, C. J. y Cootes, T. F., *An Automatic Face Identification System Using Flexible Appearance Models*. Image and Vision Computing, 13(5), 393-401, 1994. Doi: 10.1016/0262-8856(95)99726-H. URL: http://www.sciencedirect.com/science/article/pii/026288569599726H

[32] Li, Y., Gong, S. y Liddell, H., *Support vector regression and classification based multi-view face detection and recognition*. Automatic Face and Gesture Recognition, 300-305, 2000. Doi: 10.1109/AFGR.2000.840650. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=840650&isnumber=18088

[33] Liu C., *A Bayesian Discriminating Features Method for Face Detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25(6): 725-740, 2003. DOI: 10.1109/TPAMI.2003.1201822. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1201822&isnumber=27059

[34] Liu, C., *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. Massachusetts Istitute of Technology, 2009. URL: http://dspace.mit.edu/handle/1721.1/53293

[35] Liu, F. y Gleicher, M., *Region Enhanced Scale-Invariant Saliency Detection*. IEEE International Conference on Multimedia and Expo, 1477-1480, 2006. Doi: 10.1109/ICME.2006.262821. URL:

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4036890&isnumber=4036510

[36] OpenCV, *Haar Feature-based Cascade Classifier for Object Detection*. URL: http://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html

[37] Osadchy, M., Cun, Y. L. y Miller, L. M., *Synergistic Face Detection and Pose Estimation with Energy-Based Models*. Journal of Machine Learning Research, 8: 1197-1215, 2007. Doi: 10.1007/11957959_10. URL: http://link.springer.com/chapter/10.1007%2F11957959_10

[38] Osuna, E., Freund, R., y Girosit, F., *Training support vector machines: an application to face detection*. Computer Vision and Pattern Recognition, 130-136, 1997. Doi: 10.1109/CVPR.1997.609310. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=609310&isnumber=13322

[39] Papageorgiou, C., Oren, M. y Poggio, T., *A General Framework for Object Detection*. Computer Vision, 555-562, 1998. Doi: 10.1109/ICCV.1998.710772. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=710772&isnumber=15374

[40] Perazzi, F., Krahenbuhl, P., Pritch, Y. y Hornung, A., *Saliency Filters: Contrast Based Filtering for Salient Region Detection*. IEEE International Journal of Computer Vision, 733-740, 2012. Doi: 10.1109/CVPR.2012.6247743. URL: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6247743

[41] Proesmans, M., Van Gool, L, Pauwels, E., *Determination of Optical Flow and its Discontinuities Using Non-linear Difussion*, Computer Vision, 294-304, 1994. Doi: 10.1007/BFb0028362. URL: https://www.researchgate.net/publication/227129765_Determination_of_optical_flow_and_its_discontinuous_using_non-linear_diffusion

[42] Rätsch, M., Romdhani, S. y Vetter, T., *Efficient Face Detection by a Cascaded Support Vector Machine Using Haar-like Features*. Pattern Recognition, 62-70, 2004. Doi: 10.1007/978-3-540-28649-3_8. URL: http://link.springer.com/chapter/10.1007/978-3-540-28649-3_8

[43] Romdhani, S., Torr, P., Scholkopf, B. y Blake, A., *Computationally efficient face detection*. Computer Vision, 2001, 2: 695-700. Doi: 10.1109/ICCV.2001.937694. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=937694&isnumber=20294

[44] Roth D., Yang M. y Ahuja N., *A SNoW-based Face Detector*. 1999.URL: https://www.researchgate.net/publication/221619409_A_SNoW-based_face_detector

[45] Rowley, H., Baluja, S. y Kanade, T., *Neural Network-Based Face Detection.* Computer Vision and Pattern Recognition, 203-208, 1996. Doi:

10.1109/CVPR.1996.517075. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=517075&isnumber=11029

[46] Ruiz, C., *Detección bioinspirada de saliencia en secuencias de vídeo en Matlab*, Trabajo Fin de Grado de la Universidad Carlos III de Madrid, 2014.

[47] Sakai, T., Nagao, M. y Fujibayashi, S., *Line extraction and pattern detection in a photograph*. Pattern Recognition, 1(3): 233-248, 1969, ISSN 0031-3203. Doi: 10.1016/0031-3203(69)90006-5. URL: http://www.sciencedirect.com/science/article/pii/0031320369900065

[48] Samal, A. e Iyengar, P. A., *Human Face Detection Using Silhouettes*. International Journal of Patter Recognition and Artificial Intelligence 9(06): 845-867, 1995. Doi: 10.1142/S0218001495000353. URL: https://www.researchgate.net/publication/220360362_Human_Face_Detection_Using_Silhouettes

[49] Schapire, R. E. Freund, Y., Bartlett, P. y Lee, W. S., *Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods*. The Annals of Statistcs, 26(5): 1651-1686, 1998. URL: https://projecteuclid.org/euclid.aos/1024691352

[50] Schneiderman, H. y Kanade, T., *A Statistical Method for 3D Object Detection Applied to Faces and Cars*. Computer Vision and Pattern Recognition, 1: 746-751, 2000. Doi: 10.1109/CVPR.2000.855895. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=855895&isnumber=18553

[51] Shao, H., Zhang, Y., Xian, M., Cheng, H. D., Xu, F. y Ding, J, *A Saliency Model for Automated Tumor Detection in Breast Ultrasound Images*. International Conference on Image Processing, 1424-1428, 2015. Doi: 10.1109/ICIP.2015.7351035. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7351035&isnumber=7350743

[52] Sung, K. y Poggio, T., *Example-Based Learning for View-Based Human Face Detection.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(1): 39-51, 1998. Doi: 10.1109/34.655648 . URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=655648&isnumber=14286

[53] Tsukamoto, A., Lee, C. W. y Tsuji, S., *Detection and Tracking of Human Face with Synthesized Templates*, 1993. URL: https://www.researchgate.net/publication/245897775_Detection_and_Tracking_of_Human_Face_with_Synthesized_Templates

[54] Universidad Carlos III de Madrid, *Métodos de Boosting*. Departamento de Teoría de la Señal y Comunicaciones. URL: http://g2pi.tsc.uc3m.es/es/Boosting-es

[55] Viola, P. y Jones, M. J., *Robust Real-Time Face Detection*. International Journal of Computer Vision 57(2): 137-154, 2004. Doi: 10.1023/B:VISI.0000013087.49260.fb. URL: http://link.springer.com/article/10.1023%2FB%3AVISI.0000013087.49260.fb

[56] Walther, D. y Koch, C., *Modeling attention to salient proto-objects*. Neural Networks 19: 1395-1407, 2006. URL: http://bwlab.chass.utoronto.ca/wp-content/uploads/2014/10/walther_koch_neuralnetworks06.pdf

[57] Wang, Y., *An Analysis of the Viola-Jones Face Detection Algorithm*. 2014. URL: http://dx.doi.org/10.5201/ipol.2014.104

[58] Yan, J., *Ensemble SVM Regression Based Multi-View Face Detection System*. IEEE Workshop on Machine Learning for Signal Processing, 163-169, 2007. Doi: 10.1109/MLSP.2007.4414300. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4414300&isnumber=4414265

[59] Yang, M.-H., Kriegman, D. J. y Ahuja, N., *Detecting Faces in Images: A Survey*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(1), 34-58, 2002. Doi: 10.1109/34.982883. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=982883&isnumber=21178

[60] Yuille, A. L., Hallinan, P. W. y Cohen, D. S., *Feature Extraction from Faces Using Deformable Templates*. Computer Vision and Pattern Recognition, 104-109, 1989. Doi: 10.1109/CVPR.1989.37836. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=37836&isnumber=1531

[61] Zafeiriou, S., Zhang, C. y Zhang, Z., *A Survey on Face Detection in the Wild: Past, Present and Future*. Computer Vision and Image Understanding, 138, 1-24, 2015. Doi: doi:10.1016/j.cviu.2015.03.015. URL: http://www.sciencedirect.com/science/article/pii/S1077314215000727

[62] Zhang, C., Zang, Z., *Improving Multiview Face Detection with Multi-Task Deep Convolutional Neural Networks*. 2014. Doi: 10.1109/WACV.2014.6835990. URL: https://www.researchgate.net/publication/269299894_Improving_multiview_face_detection_with_multi-task_deep_convolutional_neural_networks

[63] Zhang, L., Tong, M. H., Marks, T. K., Shan, H. y Cottrell, G.W., *SUN: A Bayesian Framework for Saliency Using Natural Statistics*. Journal of Vision, 8(7): 32, 1-20, 2008. Doi: 10.1167/8.7.32. URL: http://www.ncbi.nlm.nih.gov/pubmed/19146264

[64] Zhong, S., Liu, Y., Ng, T.-Y. y Liu, Y., *Perception-Oriented Video Sliency Detection Via Spatio-Temporal Attention Analysis*. 2016. Doi: 10.1016/j.neucom.2016.04.048. URL: http://www.sciencedirect.com/science/article/pii/S0925231216303277